(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-213650
(P2004-213650A)
(43)公開日　平成16年7月29日(2004.7.29)

(51)Int.Cl.[7]	F I	テーマコード（参考）
G06F 12/14	G06F 12/14 320B	5B017

審査請求 未請求　請求項の数 12　OL　（全 34 頁）

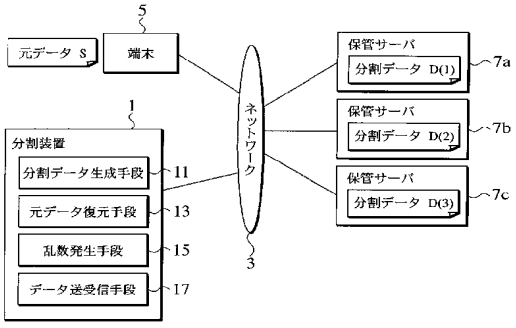
(21)出願番号　特願2003-423530(P2003-423530)
(22)出願日　平成15年12月19日(2003.12.19)
(31)優先権主張番号　特願2002-367608(P2002-367608)
(32)優先日　平成14年12月19日(2002.12.19)
(33)優先権主張国　日本国(JP)

(71)出願人　399035766
エヌ・ティ・ティ・コミュニケーションズ株式会社
東京都千代田区内幸町一丁目１番６号
(74)代理人　100083806
弁理士　三好　秀和
(74)代理人　100068342
弁理士　三好　保男
(74)代理人　100095500
弁理士　伊藤　正和
(74)代理人　100101247
弁理士　高橋　俊一
(74)代理人　100098327
弁理士　高松　俊雄

最終頁に続く

(54)【発明の名称】データ分割方法、データ分割装置およびコンピュータプログラム

(57)【要約】
【課題】　比較的簡単な処理により元データを効率的に分割し得るデータ分割方法および装置を提供する。
【解決手段】　元データS、分割数n、処理単位ビット長bを設定し、元データSを処理単位ビット長b毎に区分けして複数の元部分データS(j)を生成し、複数の乱数部分データR(j)を生成し、各分割データD(i)を構成する各分割部分データ(i,j)を元部分データと乱数部分データの排他的論理和からなる所定の定義式に従って生成する。
【選択図】　図１



【特許請求の範囲】
【請求項１】
元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割方法であって、
元データを処理単位ビット長毎に区分けして、複数の元部分データを生成し、
この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成し、
各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処理単位ビット長毎に生成し、
所望の分割数の分割データを複数の分割部分データから生成することにより、各分割データのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにする
ことを特徴とするデータ分割方法。
【請求項２】
元部分データと乱数部分データは、分割数より１つ少ない複数個ずつ生成されることを特徴とする請求項１記載のデータ分割方法。
【請求項３】
分割データは、乱数のみからなる１つ以上の分割データと、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる１つ以上の分割データを含むことを特徴とする請求項１記載のデータ分割方法。
【請求項４】
乱数のみからなる１つの分割データは、任意に定めた長さの乱数を繰り返すことによって構成されることを特徴とする請求項３記載のデータ分割方法。
【請求項５】
乱数のみからなる１つの分割データは、疑似乱数生成アルゴリズムに基づいて所定の長さの情報から生成された疑似乱数により構成されることを特徴とする請求項３または請求項４記載のデータ分割方法。
【請求項６】
分割データは、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる２つ以上の分割データを含むことを特徴とする請求項１記載のデータ分割方法。
【請求項７】
元データ、乱数、分割データ、分割数および処理単位ビット長をそれぞれS,R,D,nおよびbで表すとともに、変数としてi(=1～n)およびj(=1～n 1)を用いて複数(n 1)個の元部分データ、複数(n 1)個の乱数部分データ、複数(n)個の分割データおよび各分割データの複数(n 1)個の分割部分データのそれぞれのうちの１つをそれぞれS(j),R(j),D(i)およびD(i,j)で表わし、変数jを１からn 1まで変えて、各元部分データS(j)を元データSのb×(j 1)+1ビット目からbビット分のデータとして作成し、U[n,n]を
n×n行列でi行j列の値u(i,j)が
i+j≦n+1 のとき u(i,j)=1
i+j＞n+1 のとき u(i,j)=0
である行列とし、P[n,n]をn×n行列でi行j列の値p(i,j)が
j=i+1 のとき P(i,j)=1
i=1,j=n のとき P(i,j)=1
上記以外のとき P(i,j)=0
である行列としたとき、c(j,i,k)を(n 1)×(n 1)行列であるU[n 1,n 1]×P[n 1,n 1]^(j 1)のi行k列の値と定義し、ただしU[n 1,n 1]×P[n 1,n 1]^(j 1)とは行列U[n 1,n 1]とj 1個のP[n 1,n 1]の積を表し、Q(j,i,k)をc(j,i,k)=1のとき、Q(j,i,k)=R(k)、c(j,i,k)=0のとき、Q(j,i,k)=0 と定義したとき、各分割部分データD(i,j)を、変数iを１からnまで変えながら各変数iにおいて変数jを１からn 1まで変えて、i<nのとき、

【数１】

$$D(i,j) = S(j)*(\prod_{k=1}^{n-1} Q(j,i,k))$$

とし、i=nのとき、
D(i,j)=R(j)
として生成する、ただし
【数２】

$$\prod_{k=1}^{n-1} Q(j,i,k) = Q(j,i,1)*Q(j,i,2)*\cdots\cdots*Q(j,i,n-1)$$

*は排他的論理和演算を表す、ことを特徴とする請求項１記載のデータ分割方法。
【請求項８】
各分割データは、各分割データを構成する分割部分データ間で演算を行うことによって乱数成分が失われることがないように生成されることを特徴とする請求項１記載のデータ分割方法。
【請求項９】
各分割データは、まず元部分データと乱数部分データの排他的論理和演算による所定の定義式を用いて各分割データを構成する複数の分割部分データを生成し、その後各分割データを構成するある１つの分割部分データと別の１つの分割部分データを入れ換えることによって生成されることを特徴とする請求項８記載のデータ分割方法。
【請求項１０】
各分割データは、まず元部分データと乱数部分データの排他的論理和演算による所定の定義式を用いて各分割データD(i)を構成する複数の分割部分データD(i,j)を生成し、その後各iの値がn 1>i>0であるD(i,j)からj番目の乱数部分データR(j)
を削除することによって生成される、ただしnは所望の分割数、j=(n 1)×m+1, m≧0は任意の整数である、ことを特徴とする請求項８記載のデータ分割方法。
【請求項１１】
元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割装置であって、
元データを処理単位ビット長毎に区分けして、複数の元部分データを生成する元部分データ生成手段と、
この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成する乱数生成手段と、
各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処理単位ビット長毎に生成する分割部分データ生成手段と、
所望の分割数の分割データを複数の分割部分データから生成することにより、各分割データのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにする分割データ生成手段と
を有することを特徴とするデータ分割装置。
【請求項１２】
元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割用のコンピュータプログラムであって、
元データを処理単位ビット長毎に区分けして、複数の元部分データを生成し、
この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成し、
各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処理単位ビット長毎に生成し、
所望の分割数の分割データを複数の分割部分データから生成することにより、各分割デ

ータのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにする
ことをコンピュータに実行させることを特徴としたコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データの機密性および安全性を確保するためにはデータを分割して保管することが有効であるが、このような場合などに有効なデータ分割方法および装置に関し、更に詳しくは、元データを所望の処理単位ビットに基づいて所望の分割数の分割データに分割するデータ分割方法および装置に関する。
【背景技術】
【０００２】
重要な秘密データ（以下、元データという）を保管する場合、紛失、破壊、盗難やプライバシー侵害の脅威がある。このような脅威は秘密に保管すべきデータを単に暗号化しただけでは解決できず、紛失、破壊に備えてコピーを複数作ることが有効であるが、コピーを複数作ると盗難のリスクが増加してしまう。
【０００３】
このような問題を解決する手段として、従来、しきい値秘密分散法がある（非特許文献１参照）。この従来の方法は、元データSをn個のデータに分割し、そのうち任意のx個の分割データを集めれば元データSが復元できるが、任意のx 1個の分割データでは元データSは復元できないというものである。従って、x 1個まで分割データが盗まれても元データSが漏れず、またn x個まで分割データを紛失したり破壊されたりしても、元データSを復元できる。
【０００４】
この方法の代表的な実現例としてn 1次多項式と剰余演算により構成される方法がある（非特許文献２参照）。この従来の方法は、公開鍵暗号方式の秘密鍵の分割管理などで利用されており、データ量があまり多くないため、現状のコンピュータの演算処理能力、記憶装置・記憶媒体などのコストに対しては特に問題ない。
【非特許文献１】A. Shamir, "How to Share a Secret", Comm.Assoc.Comput.Mach., Vol. 22, no. 11, pp. 612 613(Nov.1979)
【非特許文献２】Bruce Schneier, "Applied Cryptography", John Wiley&Sons, Inc., pp. 383 384(1994)
【発明の開示】
【発明が解決しようとする課題】
【０００５】
上述した従来の方法を安全に保管したいデータ量が例えばメガバイト、ギガバイトまたはそれ以上の規模となった場合に利用すると、多項式演算・剰余演算などを含む多倍長整数の演算処理を大量のデータに対して行う演算処理能力が必要となるとともに、またこの従来の方法では、例えば分割数n=5の場合には１バイトのデータから１バイトの分割データが５つ生成されるため、元データに対して単純に分割数に比例した倍数の記憶容量が必要となるなど、コンピュータを用いて具体的に実現する上で現実的ではないという問題がある。
【０００６】
また、上述した従来の方法では、データの機密性を確保するため分割演算のための処理単位を設定しているが、この分割演算の処理単位にある程度のデータ長が必要となり、任意の処理単位で分割演算を行うことができないという問題もある。
【０００７】
本発明は、上記に鑑みてなされたもので、その目的とするところは、比較的簡単な処理により元データを効率的に分割し得るデータ分割方法、装置およびコンピュータプログラムを提供することにある。

【課題を解決するための手段】
【０００８】
上記目的を達成するため、本発明は、元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割方法であって、元データを処理単位ビット長毎に区分けして、複数の元部分データを生成し、この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成し、各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処理単位ビット長毎に生成し、所望の分割数の分割データを複数の分割部分データから生成することにより、各分割データのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにすることを特徴とするデータ分割方法を提供する。
【０００９】
また、本発明では、元部分データと乱数部分データは、分割数より１つ少ない複数個ずつ生成されることを特徴とする。
【００１０】
また、本発明では、分割データは、乱数のみからなる１つ以上の分割データと、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる１つ以上の分割データを含むことを特徴とする。
【００１１】
また、本発明では、乱数のみからなる１つの分割データは、任意に定めた長さの乱数を繰り返すことによって構成されることを特徴とする。
【００１２】
また、本発明では、乱数のみからなる１つの分割データは、疑似乱数生成アルゴリズムに基づいて所定の長さの情報から生成された疑似乱数により構成されることを特徴とする。
【００１３】
また、本発明では、分割データは、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる２つ以上の分割データを含むことを特徴とする。
【００１４】
また、本発明は、元データ、乱数、分割データ、分割数および処理単位ビット長をそれぞれS,R,D,nおよびbで表すとともに、変数としてi(=1～n)およびj(=1～n 1)を用いて複数(n 1)個の元部分データ、複数(n 1)個の乱数部分データ、複数(n)個の分割データおよび各分割データの複数(n 1)個の分割部分データのそれぞれのうちの１つをそれぞれS(j),R(j),D(i)およびD(i,j)で表わし、変数jを１からn 1まで変えて、各元部分データS(j)を元データSのb×(j 1)+1ビット目からbビット分のデータとして作成し、U[n,n]を
n×n行列でi行j列の値u(i,j)が
i+j≦n+1 のとき u(i,j)=1
i+j＞n+1 のとき u(i,j)=0
である行列とし、P[n,n]をn×n行列でi行j列の値p(i,j)が
j=i+1 のとき P(i,j)=1
i=1,j=n のとき P(i,j)=1
上記以外のとき P(i,j)=0
である行列としたとき、c(j,i,k)を(n 1)×(n 1)行列であるU[n 1,n 1]×P[n 1,n 1]^(j 1)のi行k列の値と定義し、ただしU[n 1,n 1]×P[n 1,n 1]^(j 1)とは行列U[n 1,n 1]とj 1個のP[n 1,n 1]の積を表し、Q(j,i,k)をc(j,i,k)=1のとき、Q(j,i,k)=R(k),c(j,i,k)=0のとき、Q(j,i,k)=0 と定義したとき、各分割部分データD(i,j)を、変数iを１からnまで変えながら各変数iにおいて変数jを１からn 1まで変えて、i<nのとき、

【数３】

$$D(i,j) = S(j)*(\prod_{k=1}^{n-1} Q(j,i,k))$$

【００１５】

とし、i=nのとき、

D(i,j)=R(j)

として生成する、ただし

【数４】

$$\prod_{k=1}^{n-1} Q(j,i,k) = Q(j,i,1)*Q(j,i,2)*\cdots\cdots*Q(j,i,n-1)$$

【００１６】

＊は排他的論理和演算を表す、ことを特徴とする。

【００１７】

また、本発明では、各分割データは、各分割データを構成する分割部分データ間で演算を行うことによって乱数成分が失われることがないように生成されることを特徴とする。

【００１８】

また、本発明では、各分割データは、まず元部分データと乱数部分データの排他的論理和演算による所定の定義式を用いて各分割データを構成する複数の分割部分データを生成し、その後各分割データを構成するある１つの分割部分データと別の１つの分割部分データを入れ換えることによって生成されることを特徴とする。

【００１９】

また、本発明では、各分割データは、まず元部分データと乱数部分データの排他的論理和演算による所定の定義式を用いて各分割データD(i)を構成する複数の分割部分データD(i,j)を生成し、その後各iの値がn 1>i>0であるD(i,j)からj番目の乱数部分データR(j)を削除することによって生成される、ただしnは所望の分割数、j=(n 1)×m+1, m≧0は任意の整数である、ことを特徴とする。

【００２０】

さらに、本発明は、元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割装置であって、元データを処理単位ビット長毎に区分けして、複数の元部分データを生成する元部分データ生成手段と、この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成する乱数生成手段と、各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処理単位ビット長毎に生成する分割部分データ生成手段と、所望の分割数の分割データを複数の分割部分データから生成することにより、各分割データのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにする分割データ生成手段とを有することを特徴とするデータ分割装置を提供する。

【００２１】

さらに、本発明は、元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するデータ分割用のコンピュータプログラムであって、元データを処理単位ビット長毎に区分けして、複数の元部分データを生成し、この複数の元部分データの各々に対応して、元データのビット長と同じまたはこれより短い長さの乱数から処理単位ビット長の複数の乱数部分データを生成し、各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和によって処置単位ビット長毎に生成し、所望の分割数の分割データを複数の分割部分データから生成することにより、各分割データのみから元データが不明であるが、生成した分割データのうちの所定の個数の分割データから元データが復元可能であるようにすることをコンピュータに実行させることを特徴としたコンピュータプログラムを提供する。

【発明の効果】

【００２２】

以上説明したように、本発明によれば、元データを処理単位ビット長毎に区分けして複数の元部分データを生成し、複数の乱数部分データを生成し、各分割データを構成する各分割部分データを元部分データと乱数部分データの排他的論理和からなる所定の定義式に従って生成するので、従来のように多項式や剰余演算を用いることなく、コンピュータ処理に適したビット演算である排他的論理和演算を用いることにより高速かつ高性能な演算処理能力を必要とせず、大容量のデータに対しても簡単な演算処理を繰り返して分割データを簡単かつ迅速に生成することができるとともに、また分割データの保管に必要となる記憶容量も分割数に比例した倍数の容量よりも小さくすることができる。

【発明を実施するための最良の形態】

【００２３】

以下、図面を用いて本発明の実施の形態を説明する。図１は、本発明の一実施形態に係るデータ分割方法を実施するデータ分割装置を含むシステム構成図である。本実施形態のデータ分割装置は、符号１で示すように分割装置１としてネットワーク３に接続されて設けられ、このネットワーク３にアクセスしてくる端末５からの元データ分割要求に応じて元データSを複数の分割データに分割し、この分割した複数の分割データをネットワーク３を介して複数の保管サーバ７ａ，７ｂ，７ｃに保管するようになっている。なお、図１では、分割装置１は、端末５からの元データSを３つの分割データD(1),D(2),D(3)に分割し、それぞれを複数の保管サーバ７ａ，７ｂ，７ｃに保管するようにしている。

【００２４】

また、分割装置１は、ネットワーク３を介してアクセスしてくる端末５からの元データ復元要求に応じて複数の分割データD(1),D(2),D(3)をネットワーク３を介して各保管サーバ７から取得し、この取得した複数の分割データD(1),D(2),D(3)から元データSを復元し、ネットワーク３を介して端末５に送信するようになっている。

【００２５】

分割装置１は、詳しくは、元データSから複数の分割データDを生成する分割データ生成手段１１、複数の分割データDから元データSを復元する元データ復元手段１３、元データSから複数の分割データDを生成するために使用される乱数Rを発生する乱数発生手段１５、および分割データ生成手段１１で生成した複数の分割データDをネットワーク３を介して複数の保管サーバ７ａ，７ｂ，７ｃに送信したり、また複数の保管サーバ７ａ，７ｂ，７ｃからの複数の分割データDをネットワーク３を介して受信するためのデータ送受信手段１７から構成されている。

【００２６】

上述したように構成される本実施形態における元データの分割および復元では、元データを所望の処理単位ビット長に基づいて所望の分割数の分割データに分割するが、この場合の処理単位ビット長は任意の値に設定することができ、元データを処理単位ビット長毎に区分けして、この元部分データから分割部分データを分割数より１少ない数ずつ生成するので、元データのビット長が処理単位ビット長の（分割数 1）倍の整数倍に一致しない場合は、元データの末尾の部分に０を埋めるなどして元データのビット長を処理単位ビット長の（分割数 1）倍の整数倍に合わせることにより本実施形態を適用することができる。

【００２７】

また、上述した乱数も（分割数 1）個の元部分データの各々に対応して処理単位ビット長のビット長を有する（分割数 1）個の乱数部分データとして乱数発生手段１５から生成される。すなわち、乱数は処理単位ビット長毎に区分けされて、処理単位ビット長のビット長を有する（分割数 1）個の乱数部分データとして生成される。更に、元データは処理単位ビット長に基づいて所望の分割数の分割データに分割されるが、この分割データの各々も（分割数 1）個の元部分データの各々に対応して処理単位ビット長のビット長を有する（分割数 1）個の分割部分データとして生成される。すなわち、分割データの各々は、

処理単位ビット長毎に区分けされて、処理単位ビット長のビット長を有する（分割数 1）個の分割部分データとして生成される。

【００２８】

なお、以下の説明では、上述した元データ、乱数、分割データ、分割数および処理単位ビット長をそれぞれS,R,D,nおよびbで表すとともに、また複数のデータや乱数などのうちの１つを表わす変数としてi(=1～n)およびj(=1～n 1)を用い、（分割数n 1）個の元部分データ、（分割数n 1）個の乱数部分データ、および分割数n個の分割データDのそれぞれのうちの１つをそれぞれS(j),R(j)およびD(i)で表記し、更に各分割データD(i)を構成する複数(n 1)の分割部分データをD(i,j)で表記するものとする。すなわち、S(j)は、元データSの先頭から処理単位ビット長毎に区分けして１番から順に採番した時のj番目の元部分データを表すものである。

【００２９】

この表記を用いると、元データ、乱数データ、分割データとこれらをそれぞれ構成する元部分データ、乱数部分データ、分割部分データは、次のように表記される。

【００３０】

元データS=(n 1)個の元部分データS(j)

=S(1),S(2), ,S(n 1)

乱数R=(n 1)個の乱数部分データR(j)

=R(1),R(2), ,R(n 1)

n個の分割データD(i)=D(1),D(2), ,D(n)

各分割部分データD(i,j)

=D(1,1),D(1,2), ,D(1,n 1)

D(2,1),D(2,2), ,D(2,n 1)

D(n,1),D(n,2), ,D(n,n 1)

(i=1～n),(j=1～n 1)

本実施形態は、上述したように処理単位ビット長毎に区分けされる複数の部分データに対して元部分データと乱数部分データの排他的論理和演算（XOR）を行って、詳しくは、元部分データと乱数部分データの排他的論理和演算（XOR）からなる定義式を用いて、元データの分割を行うことを特徴とするものであり、上述したデータ分割処理に多項式や剰余演算を用いる従来の方法に比較して、コンピュータ処理に適したビット演算である排他的論理和（XOR）演算を用いることにより高速かつ高性能な演算処理能力を必要とせず、大容量のデータに対しても簡単な演算処理を繰り返して分割データを生成することができるとともに、また分割データの保管に必要となる記憶容量も分割数に比例した倍数の容量よりも小さくすることができる。更に、任意に定めた一定の長さ毎にデータの先頭から順に演算処理を行うストリーム処理により分割データが生成される。

【００３１】

なお、本実施形態で使用する排他的論理和演算（XOR）は、以下の説明では、「＊」なる演算記号で表すことにするが、この排他的論理和演算のビット毎の演算規則での各演算結果は下記のとおりである。

【００３２】

０＊０の演算結果は ０

０＊１の演算結果は １

１＊０の演算結果は １

１＊１の演算結果は ０

また、XOR演算は交換法則、結合法則が成り立つ。すなわち、

a*b=b*a

(a*b)*c=a*(b*c)

が成り立つことが数学的に証明される。

【００３３】

また、a*a=0, a*0=0*a=aが成り立つ。
【００８４】
ここでa, b, cは同じ長さのビット列を表し、0はこれらと同じ長さですべて「0」からなるビット列を表す。
【００８５】
次に、フローチャートなどの図面も参照して、上記実施形態の作用について説明するが、この説明の前に図２、図５、図８、図９のフローチャートに示す記号の定義について説明する。
【００８６】
（１）
【数５】

$$(1)\ \prod_{i=1}^{n} A(i)$$

【００８７】
は、A(1)*A(2)* A(n)を意味するものとする。
【００８８】
（２）c(j, i, k)を、(n 1)×(n 1)行列であるU[n 1, n 1]×(P[n 1, n 1])^(j 1)のi行k列の値と定義する。
【００８９】
このときQ(j, i, k)を下記のように定義する。
【００４０】
c(j, i, k)=1 のとき Q(j, i, k)=R((n 1)×m+k)
c(j, i, k)=0 のとき Q(j, i, k)=0
ただし、mはm≧０の整数を表す。
【００４１】
（３）U[n, n]とは、n×n行列であって、i行j列の値をu(i, j)で表すと、
i+j≦n+1 のとき u(i, j)=1
i+j＞n+1 のとき u(i, j)=0
である行列を意味するものとし、「上三角行列」ということとする。具体的には下記のような行列である。
【数６】

$$U[3,3] = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$U[4,4] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

【００４２】
（４）P[n, n]とは、n×n行列であって、i行j列の値をp(i, j)で表すと、
j=i+1 のとき p(i, j)=1
i=1, j=n のとき p(i, j)=1
上記以外のとき p(i, j)=0
である行列を意味するものとし、「回転行列」ということとする。具体的には下記のよう

な行列であり、他の行列の右側からかけると当該他の行列の１列目を２列目へ、２列目を３列目へ、 ,n 1列目をn列目へ、n列目を１列目へ移動させる作用がある。つまり、行列Pを他の行列に右側から複数回かけると、その回数分だけ各列を右方向へ回転させるように移動させることができる。

【数７】

$$P[3,3] = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$P[4,4] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

【００４３】

（５）A,Bをn×n行列とすると、A×Bとは行列AとBの積を意味するものとする。行列の成分同士の計算規則は通常の数学で用いるものと同じである。

【００４４】

（６）Aをn×n行列とし、iを整数とすると、A＾iとは行列Aのi個の積を意味するものとする。また、A＾0とは単位行列Eを意味するものとする。

【００４５】

（７）単位行列E[n,n]とは、n×n行列であって、i行j列の値をe(i,j)で表すと、

i=j　　のとき e(i,j)=1

上記以外のとき e(i,j)=0

である行列を意味するものとする。具体的には下記のような行列である。Aを任意のn×n行列とすると

A×E=E×A=A

となる性質がある。

【数８】

$$E[3,3] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$E[4,4] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

【００４６】

次に、図２に示すフローチャートおよび図３、図４に示す具体的データなどを参照して、上記実施形態の作用として、まず元データSの分割処理について説明する。

【００４７】

本実施形態の分割装置１の利用者は、端末５からネットワーク３を介して分割装置１にアクセスし、分割装置１に元データSを送信し、分割装置１ではデータ送受信手段１７が端末５からの元データSを受信し、分割装置１に供給する（図２のステップＳ２０１）。なお、本例では、元データSは、１６ビットの「10110010 00110111」とする。

【００４８】
次に、利用者は端末５から分割数nとして３を分割装置１に指示する（ステップＳ２０３）。この分割数nは分割装置１において予め定められた値を用いてもよい。なお、この分割数n=3に従って分割装置１で生成される３個の分割データをD(1),D(2),D(3)とする。この分割データD(1),D(2),D(3)は、すべて元データのビット長と同じ１６ビット長のデータである。
【００４９】
それから、元データSを分割するために使用される処理単位ビット長bを８ビットと決定し、また元データと同じビット長である１６ビットの乱数Rを乱数発生手段１５から取得して生成する（ステップＳ２０５）。この処理単位ビット長bは、利用者が端末５から分割装置１に対して指定してもよいし、または分割装置１において予め定められた値を用いてもよい。なお、処理単位ビット長bは、任意のビット数でよいが、ここでは元データSを割り切れることができる８ビットとしている。従って、上記１６ビットの「10110010 00110111」の元データSは、８ビットの処理単位ビット長で区分けされた場合の２個の元分割データS(1)およびS(2)は、それぞれ「10110010」および「00110111」となる。
【００５０】
次のステップＳ２０７では、元データSのビット長が8×2の整数倍であるか否かを判定し、整数倍でない場合には、元データSの末尾を０で埋めて、8×2の整数倍に合わせる。なお、本例のように処理単位ビット長bが８ビットおよび分割数nが３に設定された場合における分割処理は、元データSのビット長として１６ビットに限られるものでなく、処理単位ビット長b×(分割数n 1)=8×2の整数倍の元データSに対して有効なものである。
【００５１】
次に、ステップＳ２０９では、変数m、すなわち上述した整数倍を意味する変数mを０に設定する。本例のように、元データSが処理単位ビット長b×(分割数n 1)=8×2=16ビットである場合には、変数mは０であるが、２倍の３２ビットの場合には、変数mは１となり、３倍の４８ビットの場合には、変数mは２となる。
【００５２】
次に、元データSの8×2×m+1ビット目から8×2ビット分のデータが存在するか否かが判定される（ステップＳ２１１）。これは、このステップＳ２１１以降に示す分割処理を元データSの変数mで特定される処理単位ビット長b×(分割数n 1)=8×2=16ビットに対して行った後、元データSとして次の１６ビットがあるか否かを判定しているものである。本例のように元データSが１６ビットである場合には、１６ビットの元データSに対してステップＳ２１１以降の分割処理を１回行うと、後述するステップＳ２１９で変数mが+1されるが、本例の元データSでは変数mがm+1の場合に相当する１７ビット以降のデータは存在しないので、ステップＳ２１１からステップＳ２２１に進むことになるが、今の場合は、変数mは０であるので、元データSの8×2×m+1ビット目は、8×2×0+1=1となり、元データSの１６ビットの１ビット目から8×2ビット分にデータが存在するため、ステップＳ２１３に進む。
【００５３】
ステップＳ２１３では、変数jを１から２(=分割数n 1)まで変えて、元データSの8×(2×m+j 1)+1ビット目から８ビット分(=処理単位ビット長)のデータを元部分データS(2×m+j)に設定し、これにより元データSを処理単位ビット長で区分けした2(分割数n 1)個の元部分データS(1),S(2)を次のように生成する。
【００５４】
元データS=S(1),S(2)
第１の元部分データS(1)=「10110010」
第２の元部分データS(2)=「00110111」
次に、変数jを１から２(=分割数n 1)まで変えて、乱数部分データR(2×m+j)に乱数発生手段１５から発生する８ビットの長さの乱数を設定し、これにより乱数Rを処理単位ビット長で区分けした2(分割数n 1)個の乱数部分データR(1),R(2)を次のように生成する（ス

テップＳ２１５）。

【００５５】

乱数R=R(1),R(2)

第１の乱数部分データR(1)=「10110001」

第２の乱数部分データR(2)=「00110101」

次に、ステップＳ２１７において、変数iを１から３(=分割数n)まで変えるとともに、更に各変数iにおいて変数jを１から２(=分割数n 1)まで変えながら、ステップＳ２１７に示す分割データを生成するための元部分データと乱数部分データの排他的論理和からなる定義式により複数の分割データD(i)の各々を構成する各分割部分データD(i,2×m+j) を生成する。この結果、次に示すような分割データDが生成される。

【００５６】

分割データD

=３個の分割データD(i)=D(1),D(2),D(3)

第１の分割データD(1)

=２個の分割部分データD(1,j)=D(1,1),D(1,2)

=「00110110」,「10110011」

第２の分割データD(2)

=２個の分割部分データD(2,j)=D(2,1),D(2,2)

=「00000011」,「00000010」

第３の分割データD(3)

=２個の分割部分データD(3,j)=D(3,1),D(3,2)

=「10110001」,「00110101」

なお、各分割部分データD(i,j)を生成するためのステップＳ２１７に示す定義式は、本例のように分割数n=3の場合には、具体的には図４に示す表に記載されているものとなる。図４に示す表から、分割部分データD(1,1)を生成するための定義式はS(1)*R(1)*R(2)であり、D(1,2)の定義式はS(2)*R(1)*R(2)であり、D(2,1)の定義式はS(1)*R(1)であり、D(2,2)の定義式はS(2)*R(2)であり、D(3,1)の定義式はR(1)であり、D(3,2)の定義式はR(2)である。また、図４に示す表にはm＞0の場合の任意の整数についての一般的な定義式も記載されている。

【００５７】

このように整数倍を意味する変数m=0の場合について分割データDを生成した後、次に変数mを１増やし（ステップＳ２１９）、ステップＳ２１１に戻り、変数m+1に該当する元データSの１７ビット以降について同様の分割処理を行おうとするが、本例の元データSは１６ビットであり、１７ビット以降のデータは存在しないので、ステップＳ２１１からステップＳ２２１に進み、上述したように生成した分割データD(1),D(2),D(3)を分割装置１のデータ送受信手段１７からネットワーク３を介して保管サーバ７ａ，７ｂ，７ｃにそれぞれ送信し、各保管サーバ７に保管し、分割処理を終了する。

【００５８】

ここで、上述した図２のフローチャートのステップＳ２１７における定義式による分割データの生成処理、具体的には分割数n=3の場合の分割データの生成処理について詳しく説明する。

【００５９】

まず、整数倍を意味する変数m=0の場合には、ステップＳ２１７に示す定義式から各分割データD(i)=D(1)～D(3)の各々を構成する各分割部分データD(i,2×m+j)=D(i,j)(i=1～3,j=1～2)は、次のようになる。

【００６０】

D(1,1)=S(1)*Q(1,1,1)*Q(1,1,2)

D(1,2)=S(2)*Q(2,1,1)*Q(2,1,2)

D(2,1)=S(1)*Q(1,2,1)*Q(1,2,2)

D(2,2)=S(2)*Q(2,2,1)*Q(2,2,2)

D(3,1)=R(1)
D(3,2)=R(2)
上記の６つの式のうち上から４つの式に含まれるQ(j,i,k)を具体的に求める。
【００６１】
これはc(j,i,k)を2×2行列であるU[2,2]×(P[2,2])^(j 1)のi行k列の値としたとき下記のように定義される。
【００６２】
c(j,i,k)=1 のとき Q(j,i,k)=R(k)
c(j,i,k)=0 のとき Q(j,i,k)=0
ここで、
j=1のときは
【数９】

$$U[2,2]\times(P[2,2])^{(j-1)} = U[2,2]\times(P[2,2])^{0} = U[2,2]\times E[2,2] = U[2,2] = \begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}$$

【００６３】
j=2のときは
【数１０】

$$U[2,2]\times(P[2,2])^{(j-1)} = U[2,2]\times(P[2,2])^{1} = U[2,2]\times P[2,2] = \begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}\times\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$$

【００６４】
これを用いると、各分割部分データD(i,j)は次のような定義式により生成される。
【００６５】
D(1,1)=S(1)*Q(1,1,1)*Q(1,1,2)=S(1)*R(1)*R(2)
D(1,2)=S(2)*Q(2,1,1)*Q(2,1,2)=S(2)*R(1)*R(2)
D(2,1)=S(1)*Q(1,2,1)*Q(1,2,2)=S(1)*R(1)*0=S(1)*R(1)
D(2,2)=S(2)*Q(2,2,1)*Q(2,2,2)=S(2)*0*R(2)=S(2)*R(2)
上述した各分割部分データD(i,j)を生成するための定義式は、図３にも図示されている。
【００６６】
図３は、上述したように１６ビットの元データSを８ビットの処理単位ビット長に基づいて分割数n=3で３分割する場合の各データと定義式および各分割部分データから元データを復元する場合の計算式などを示す表である。
【００６７】
ここで、上述した定義式により分割データD(1),D(2),D(3)および各分割部分データD(1,1),D(1,2),D(2,1),D(2,2),D(3,1),D(3,2)を生成する過程と定義式の一般形について説明する。
【００６８】

まず、第１の分割データD(1)に対しては、第１の分割部分データD(1,1)は、上述した定義式S(1)*R(1)*R(2)で定義され、第２の分割部分データD(1,2)は定義式S(2)*R(1)*R(2)で定義される。なお、この定義式の一般形は、D(1,j)に対してはS(j)*R(j)*R(j+1)であり、D(1,j+1)に対してS(j+1)*R(j)*R(j+1)である（jは奇数とする）。定義式に従って計算すると、D(1,1)は00110110，D(1,2)は10110011となるので、D(1)は00110110 10110011である。なお、定義式の一般形は、図４にまとめて示されている。

【００６９】

また、第２の分割データD(2)に対しては、D(2,1)はS(1)*R(1)で定義され、D(2,2)はS(2)*R(2)で定義される。この定義式の一般形は、D(2,j)に対してはS(j)*R(j)であり、D(2,j+1)に対してはS(j+1)*R(j+1)である（jは奇数とする）。定義式に従って計算すると、D(2,1)は00000011，D(2,2)は00000010となるので、D(2)は00000011 00000010である。

【００７０】

更に第３の分割データD(3)に対しては、D(3,1)はR(1)で定義され、D(3,2)はR(2)で定義される。この定義式の一般形は、D(3,j)に対してはR(j)であり、D(3,j+1)に対してはR(3,j+1)である（jは奇数とする）。定義式に従って計算すると、D(3,1)は10110001，D(3,2)は00110101となるので、D(3)は10110001 00110101である。

【００７１】

上記説明は、S,R,D(1),D(2),D(3)の長さを１６ビットとしたが、データの先頭から上記分割処理を繰り返すことにより、どのような長さの元データSからでも分割データD(1),D(2),D(3)を生成することができる。また、処理単位ビット長b
は任意にとることができ、元データSの先頭から順にb×2の長さ毎に上記分割処理を繰り返すことにより任意の長さの元データ、具体的には処理単位ビット長b×2の整数倍の長さの元データに対して適用することができる。なお、元データSの長さが処理単位ビット長b×2の整数倍でない場合は、例えば、データ末尾の部分を０で埋めるなどして元データSの長さを処理単位ビット長b×2の整数倍に合わせることにより上述した本実施形態の分割処理を適用することができる。

【００７２】

次に、図３の右側に示す表を参照して、分割データから元データを復元する処理について説明する。

【００７３】

まず、利用者は端末５からネットワーク３を介して分割装置１にアクセスし、分割装置１のデータ送受信手段１７を介して元データSの復元を要求する。分割装置１は、この元データSの復元要求を受け取ると、この元データSに対応する分割データD(1),D(2),D(3)が保管サーバ７ａ，７ｂ，７ｃに保管されていることを記憶しているので、ネットワーク３を介して保管サーバ７ａ，７ｂ，７ｃから分割データD(1),D(2),D(3)を取得し、この取得した分割データD(1),D(2),D(3)から次に示すように元データSを復元する。

【００７４】

まず、分割部分データD(2,1),D(3,1)から第１の元部分データS(1)を次のように生成することができる。

【００７５】

$$
\begin{aligned}
D(2,1)*D(3,1)&=(S(1)*R(1))*R(1)\\
&=S(1)*(R(1)*R(1))\\
&=S(1)*0\\
&=S(1)
\end{aligned}
$$

具体的に計算すると、D(2,1)は00000011，D(3,1)は10110001なので、S(1)は10110010となる。

【００７６】

また、別の分割部分データから次のように第２の元部分データS(2)を生成することができる。

【００７７】

D(2,2)*D(3,2)=(S(2)*R(2))*R(2)
=S(2)*(R(2)*R(2))
=S(2)*0
=S(2)

具体的に計算すると、D(2,2)は00000010，D(3,2)は00110101なので、S(2)は00110111となる。

【００７８】

一般に、jを奇数として、

D(2,j)*D(3,j)=(S(j)*R(j))*R(j)
=S(j)*(R(j)*R(j))
=S(j)*0
=S(j)

であるから、D(2,j)*D(3,j)を計算すれば、S(j)が求まる。

【００７９】

また、一般に、jを奇数として、

D(2,j+1)*D(3,j+1)=(S(j+1)*R(j+1))*R(j+1)
=S(j+1)*(R(j+1)*R(j+1))
=S(j+1)*0
=S(j+1)

であるから、D(2,j+1)*D(3,j+1)を計算すれば、S(j+1)が求まる。

【００８０】

次に、D(1),D(3)を取得してSを復元する場合には、次のようになる。

【００８１】

D(1,1)*D(3,1)*D(3,2)=(S(1)*R(1)*R(2))*R(1)*R(2)
=S(1)*(R(1)*R(1))*(R(2)*R(2))
=S(1)*0*0
=S(1)

であるから、D(1,1)*D(3,1)*D(3,2)を計算すれば、S(1)が求まる。具体的に計算すると、D(1,1)は00110110，D(3,1)は10110001，D(3,2)は00110101なので、S(1)は10110010となる。

【００８２】

また同様に、

D(1,2)*D(3,1)*D(3,2)=(S(2)*R(1)*R(2))*R(1)*R(2)
=S(2)*(R(1)*R(1))*(R(2)*R(2))
=S(2)*0*0
=S(2)

であるから、D(1,2)*D(3,1)*D(3,2)を計算すれば、S(2)が求まる。具体的に計算すると、D(1,2)は10110011，D(3,1)は10110001，D(3,2)は00110101なので、S(2)は00110111となる。

【００８３】

一般に、jを奇数として、

D(1,j)*D(3,j)*D(3,j+1)=(S(j)*R(j)*R(j+1))*R(j)*R(j+1)
=S(j)*(R(j)*R(j))*(R(j+1)*R(j+1))
=S(j)*0*0
=S(j)

であるから、D(1,j)*D(3,j)*D(3,j+1)を計算すれば、S(j)が求まる。

【００８４】

また、一般に、jを奇数として、

D(1,j+1)*D(3,j)*D(3,j+1)=(S(j+1)*R(j)*R(j+1))*R(j)*R(j+1)
=S(j+1)*(R(j)*R(j))*(R(j+1)*R(j+1))

=S(j+1)*0*0
=S(j+1)

であるから、D(1,j+1)*D(8,j)*D(8,j+1)を計算すれば、S(j+1)が求まる。

【００８５】

次に、D(1),D(2)を取得してSを復元する場合には、次のようになる。

【００８６】

D(1,1)*D(2,1)=(S(1)*R(1)*R(2))*(S(1)*R(1))
=(S(1)*S(1))*(R(1)*R(1))*R(2)
=0*0*R(2)
=R(2)

であるから、D(1,1)*D(2,1)を計算すれば、R(2)が求まる。具体的に計算すると、D(1,1)は00110110，D(2,1)は00000011なので、R(2)は00110101となる。

【００８７】

また同様に、

D(1,2)*D(2,2)=(S(2)*R(1)*R(2))*(S(2)*R(2))
=(S(2)*S(2))*R(1)*(R(2)*R(2))
=0*R(1)*0
=R(1)

であるから、D(1,2)*D(2,2)を計算すれば、R(1)が求まる。具体的に計算すると、D(1,2)は10110011，D(2,2)は00000010なので、R(1)は10110001となる。

【００８８】

このR(1),R(2)を使用してS(1),S(2)を求める。

【００８９】

D(2,1)*R(1)=(S(1)*R(1))*R(1)
=S(1)*(R(1)*R(1))
=S(1)*0
=S(1)

であるから、D(2,1)*R(1)を計算すれば、S(1)が求まる。具体的に計算すると、D(2,1)は00000011，R(1)は10110001なので、S(1)は10110010となる。

【００９０】

また同様に、

D(2,2)*R(2)=(S(2)*R(2))*R(2)
=S(2)*(R(2)*R(2))
=S(2)*0
=S(2)

であるからD(2,2)*R(2)を計算すればS(2)が求まる。具体的に計算するとD(2,2)は00000010，R(2)は00110101なので、S(2)は00110111となる。

【００９１】

一般に、jを奇数として、

D(1,j)*D(2,j)=(S(j)*R(j)*R(j+1))*(S(j)*R(j))
=(S(j)*S(j))*(R(j)*R(j))*R(j+1)
=0*0*R(j+1)
=R(j+1)

であるからD(1,j)*D(2,j)を計算すればR(j+1)が求まる。

【００９２】

また同様に、

D(1,j+1)*D(2,j+1)=(S(j+1)*R(j)*R(j+1))*(S(j+1)*R(j+1))
=(S(j+1)*S(j+1))*R(j)*(R(j+1)*R(j+1))
=0*R(j)*0
=R(j)

であるからD(1,j+1)*D(2,j+1)を計算すればR(j)が求まる。

【００９３】

このR(j),R(j+1)を使用してS(j),S(j+1)を求める。

【００９４】

D(2,j)*R(j)=(S(j)*R(j))*R(j)
=S(j)*(R(j)*R(j))
=S(j)*0
=S(j)

であるからD(2,j)*R(j)を計算すればS(j)が求まる。

【００９５】

また同様に、

D(2,j+1)*R(j+1)=(S(j+1)*R(j+1))*R(j+1)
=S(j+1)*(R(j+1)*R(j+1))
=S(j+1)*0
=S(j+1)

であるからD(2,j+1)*R(j+1)を計算すればS(j+1)が求まる。

【００９６】

上述したように、元データの先頭から処理単位ビット長bに基づいて分割処理を繰り返し行って、分割データを生成した場合には、３つの分割データD(1),D(2),D(3)のすべてを用いなくても、３つの分割データのうち、２つの分割データを用いて上述したように元データを復元することができる。

【００９７】

本発明の他の実施形態として、乱数Rのビット長を元データSのビット長よりも短いものを使用して、元データの分割処理を行うことができる。

【００９８】

すなわち、上述した乱数RはS,D(1),D(2),D(3)と同じビット長のデータとしたが、乱数Rを元データSのビット長より短いものとし、分割データD(1),D(2),D(3)の生成にこの短いビット長の乱数Rを繰り返し用いるものである。

【００９９】

なお、分割データD(3)は乱数Rのみから生成されるので、分割データD(3)は乱数Rを繰り返して保管しておく必要はない。例えば、元データSのビット長を１６００ビット（200バイト）としたとき、乱数Rは任意にとった１６０ビット（20バイト）のデータの繰り返しとする。つまり、R(1)～R(20)はランダムに生成し、R(21)～R(200)はR(21)=R(1),R(22)=R(2), ,R(40)=R(20),R(41)=R(1),R(42)=R(2), ,R(60)=R(20),R(61)=R(1),R(62)=R(2), ,R(80)=R(20), ,R(181)=R(1),R(182)=R(2), ,R(200)=R(20)とする。

【０１００】

先の説明では、分割部分データD(3,j)を乱数部分データR(j)と定義してD(3)を生成しているが、D(3,20)まで保管すれば十分である。つまり、D(3)の長さはD(1),D(2)の１０分の１となる。従って、保管すべきデータの総量は先の実施形態では元データSの３倍であるが、この実施形態では２．１倍とすることができる。乱数Rにおける繰り返し部分のデータの長さは、短すぎると、D(1)またはD(2)のみからRが解読されてしまうことも考えられるため、適切な長さを選択することが望ましい。

【０１０１】

この実施形態では例えば乱数Ｒを生成するために疑似乱数生成アルゴリズムを使用する。乱数には自然界の物理現象などを使って乱数を発生させる真性乱数と、コンピュータのアルゴリズムなどで乱数を発生させる疑似乱数がある、真性乱数は、サイコロを何回も振ったり、雑音などの物理現象をりようしたりして生成することができるが、手間や装置がたいへんであるため、その代わりに、適当な長さの種（乱数生成の種となる情報（ｓｅｅｄｓ））から決定的なアルゴリズムに基づいて生成される疑似乱数が用いられる。例えば短い乱数を種とすれば長い乱数を得ることができる。種の長さは、例えば１２８ビット、

１６０ビットまたはそれ以上のものがある。決定的なアルゴリズムに基づいて生成されるといっても、統計的一様性、無相関性など乱数としてひつような性質を一定のレベルで満たしている。具体例としては、ＡＮＳＩ　Ｘ９．４２、ＦＩＰＳ　１８６－２など標準化されたものがある
（http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/cryptrec20030425-spec01.html）
。

【０１０２】

これらをもちいれば、乱数生成の種を入力として長い疑似乱数の列を生成することができる。例えば、１６０ビットの種を与えて元データＳのビット長と同じ長さの乱数Ｒを生成し、上述したようにしてＳとＲからD(1),D(2)を生成し、D(3)にはＲを格納するのではなく１６０ビットの種を格納して保管することにより、元データＳのビット長が大きくなってもD(3)に格納して保管すべきビット数は１６０ビットで済み、保管すべきデータの総量を押さえることができる。元データＳを復元する場合には、D(3)に格納された１６０ビットの種から元データＳのビット長と同じ長さの乱数Ｒを再度生成し、上述したようにして、これとD(1)またはD(2)を用いて元データＳを復元することができる。

【０１０３】

上述した各実施形態は、元データを３つに分割し、そのうち２つから元データが復元可能となるものであったが、分割数nを３より大きくとって、n個より少ない個数の分割データから元データを復元することができることは勿論のことである。

【０１０４】

その１つの応用例として、元データを４つの分割データに分割する分割数n=4の場合の分割処理について図５に示すフローチャートおよび図６に示す定義式の一般形などを参照して説明する。

【０１０５】

まず、利用者は端末５から分割装置１にアクセスして元データSを送信し、分割装置１ではデータ送受信手段１７が端末５からの元データSを受信し、分割装置１に供給する（ステップＳ３０１）。それから、利用者は端末５から分割数nとして４を分割装置１に指示する（ステップＳ３０３）。この分割数nは分割装置１において予め定められた値を用いてもよい。また、処理単位ビット長bが一例として８ビットと決定される（ステップＳ３０５）。次に、元データSのビット長が8×3の整数倍であるか否かを判定し、整数倍でない場合には、元データSの末尾を０で埋める（ステップＳ３０７）。また、整数倍を意味する変数mを０に設定する（ステップＳ３０９）。

【０１０６】

次に、元データSの8×3×m+1ビット目から8×3ビット分のデータが存在するか否かが判定される（ステップＳ３１１）。なお、本例では、元データSが8×3=24ビット長のデータの場合について説明している。

【０１０７】

ステップＳ３１１の判定の結果、本例の元データSでは8×3=24ビットのデータであり、変数m=1の場合に相当する8×3×m(=1)+1ビットに相当する２５ビット以降のデータは存在しないので、ステップＳ３１１からステップＳ３２１に進むことになるが、今の場合は、変数mは０であるので、元データSの8×3×m+1ビット目は、8×3×0+1=1となり、元データSの２４ビットの１ビット目から8×3ビット分にデータが存在するため、ステップＳ３１３に進む。

【０１０８】

ステップＳ３１３では、変数jを１から３（=分割数n 1）まで変えて、元データSの8×(3×m+j 1)+1ビット目から８ビット分（=処理単位ビット長）のデータを元部分データS(3×m+j)に設定し、これにより元データSを処理単位ビット長で区分けした３個の元部分データS(1),S(2),S(3)が生成される。

【０１０９】

次に、変数jを１から３まで変えて、乱数部分データR(3×m+j)に乱数発生手段１５から

発生する８ビットの長さの乱数を設定し、これにより乱数Rを処理単位ビット長で区分けした３個の乱数部分データR(1),R(2),R(3)が生成される（ステップＳ３１５）。

【０１１０】

次に、ステップＳ３１７において、変数iを１から４(=分割数n)まで変えるとともに、更に各変数iにおいて変数jを１から３(=分割数ｎ１)まで変えながら、ステップＳ３１７に示す分割データを生成するための定義式により複数の分割データD(i)の各々を構成する各分割部分データD(i,3×m+j)を生成する。この結果、次に示すような分割データDが生成される。

【０１１１】

分割データD
=4個の分割データD(i)=D(1),D(2),D(3),D(4)
第１の分割データD(1)
=3個の分割部分データD(1,j)=D(1,1),D(1,2),D(1,3)
第２の分割データD(2)
=3個の分割部分データD(2,j)=D(2,1),D(2,2),D(2,3)
第３の分割データD(3)
=3個の分割部分データD(3,j)=D(3,1),D(3,2),D(3,3)
第４の分割データD(4)
=3個の分割部分データD(4,j)=D(4,1),D(4,2),D(4,3)

なお、各分割部分データD(i,j)を生成するためのステップＳ３１７に示す定義式は、本例のように分割数n=4の場合には、具体的には図６に示す表に記載されているものとなる。図６に示す表から、分割部分データD(1,1)を生成するための定義式はS(1)*R(1)*R(2)*R(3)であり、D(1,2)の定義式はS(2)*R(1)*R(2)*R(3)であり、D(1,3)の定義式はS(3)*R(1)*R(2)*R(3)であり、D(2,1)の定義式はS(1)*R(1)*R(2)であり、D(2,2)の定義式はS(2)*R(2)*R(3)であり、D(2,3)の定義式はS(3)*R(1)*R(3)であり、D(3,1)の定義式はS(1)*R(1)であり、D(3,2)の定義式はS(2)*R(2)であり、D(3,3)の定義式はS(3)*R(3)であり、D(4,1)の定義式はR(1)であり、D(4,2)の定義式はR(2)であり、D(4,3)の定義式はR(3)である。また、図６に示す表にはm＞0の場合の任意の整数についての一般的な定義式も記載されている。

【０１１２】

このように変数m=0の場合について分割データDを生成した後、次に変数mを１増やし（ステップＳ３１９）、ステップＳ３１１に戻り、変数m=1に該当する元データSの２５ビット以降について同様の分割処理を行おうとするが、本例の元データSは２４ビットであり、２５ビット以降のデータは存在しないので、ステップＳ３１１からステップＳ３２１に進み、上述したように生成した分割データD(1),D(2),D(3),D(4)を分割装置１のデータ送受信手段１７からネットワーク３を介して保管サーバ７にそれぞれ送信し、各保管サーバ７に保管し、分割処理を終了する。図１では保管サーバは３個であるが、分割数に応じて保管サーバを増やし、各分割データを異なる保管サーバに保管することが望ましい。

【０１１３】

ここで、上述した図５のフローチャートのステップＳ３１７における定義式による分割データの生成処理、具体的には分割数n=4の場合の分割データの生成処理について詳しく説明する。

【０１１４】

まず、ステップＳ３１７に示す定義式から各分割データD(i)=D(1)～D(4)の各々を構成する各分割部分データD(i,3×m+j)は、次のようになる。

【０１１５】

D(1,3×m+1)=S(3×m+1)*Q(1,1,1)*Q(1,1,2)*Q(1,1,3)
D(1,3×m+2)=S(3×m+2)*Q(2,1,1)*Q(2,1,2)*Q(2,1,3)
D(1,3×m+3)=S(3×m+3)*Q(3,1,1)*Q(3,1,2)*Q(3,1,3)
D(2,3×m+1)=S(3×m+1)*Q(1,2,1)*Q(1,2,2)*Q(1,2,3)
D(2,3×m+2)=S(3×m+2)*Q(2,2,1)*Q(2,2,2)*Q(2,2,3)

D(2,3×m+3)=S(3×m+3)*Q(3,2,1)*Q(3,2,2)*Q(3,2,3)
D(3,3×m+1)=S(3×m+1)*Q(1,3,1)*Q(1,3,2)*Q(1,3,3)
D(3,3×m+2)=S(3×m+2)*Q(2,3,1)*Q(2,3,2)*Q(2,3,3)
D(3,3×m+3)=S(3×m+3)*Q(3,3,1)*Q(3,3,2)*Q(3,3,3)
D(4,3×m+1)=R(3×m+1)
D(4,3×m+2)=R(3×m+2)
D(4,3×m+3)=R(3×m+3)

次に、Q(j,i,k)を具体的に求める。これはc(j,i,k)を3×3行列であるU[3,3]×(P[3,3])^(j-1)のi行k列の値としたとき下記のように定義される。

【0116】

c(j,i,k)=1 のとき Q(j,i,k)=R(3×m+k)
c(j,i,k)=0 のとき Q(j,i,k)=0

j=1のときは

【数11】

$$U[3,3]\times(P[3,3])^{(j-1)} = U[3,3]\times(P[3,3])^{0} = U[3,3] = \begin{pmatrix}1&1&1\\1&1&0\\1&0&0\end{pmatrix}$$

【0117】

j=2のときは

【数12】

$$U[3,3]\times(P[3,3])^{(j-1)} = U[3,3]\times(P[3,3])^{1} = U[3,3]\times(P[3,3]) = \begin{pmatrix}1&1&1\\1&1&0\\1&0&0\end{pmatrix}\times\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix} = \begin{pmatrix}1&1&1\\0&1&1\\0&1&0\end{pmatrix}$$

【0118】

j=3のときは

【数13】

$$U[3,3]\times(P[3,3])^{(j-1)} = U[3,3]\times(P[3,3])^{2} = U[3,3]\times(P[3,3])\times(P[3,3]) = \begin{pmatrix}1&1&1\\0&1&1\\0&1&0\end{pmatrix}\times\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix} = \begin{pmatrix}1&1&1\\1&0&1\\0&0&1\end{pmatrix}$$

【0119】

これを用いると、各分割部分データは次のような定義式により生成される。

【０１２０】

D(1,3×m+1)=S(3×m+1)*Q(1,1,1)*Q(1,1,2)*Q(1,1,3)
=S(3×m+1)*R(3×m+1)*R(3×m+2)*R(3×m+3)
D(1,3×m+2)=S(3×m+2)*Q(2,1,1)*Q(2,1,2)*Q(2,1,3)
=S(3×m+2)*R(3×m+1)*R(3×m+2)*R(3×m+3)
D(1,3×m+3)=S(3×m+3)*Q(3,1,1)*Q(3,1,2)*Q(3,1,3)
=S(3×m+3)*R(3×m+1)*R(3×m+2)*R(3×m+3)
D(2,3×m+1)=S(3×m+1)*Q(1,2,1)*Q(1,2,2)*Q(1,2,3)
=S(3×m+1)*R(3×m+1)*R(3×m+2)
D(2,3×m+2)=S(3×m+2)*Q(2,2,1)*Q(2,2,2)*Q(2,2,3)
=S(3×m+2)*R(3×m+2)*R(3×m+3)
D(2,3×m+3)=S(3×m+3)*Q(3,2,1)*Q(3,2,2)*Q(3,2,3)
=S(3×m+3)*R(3×m+1)*R(3×m+3)
D(3,3×m+1)=S(3×m+1)*Q(1,3,1)*Q(1,3,2)*Q(1,3,3)
=S(3×m+1)*R(3×m+1)
D(3,3×m+2)=S(3×m+2)*Q(2,3,1)*Q(2,3,2)*Q(2,3,3)
=S(3×m+2)*R(3×m+2)
D(3,3×m+3)=S(3×m+3)*Q(3,3,1)*Q(3,3,2)*Q(3,3,3)
=S(3×m+3)*R(3×m+3)
D(4,3×m+1)=R(3×m+1)
D(4,3×m+2)=R(3×m+2)
D(4,3×m+3)=R(3×m+3)

ここで、上述したように図２のステップＳ２１７や図５のステップＳ３１７で示した定義式に基づいて元データを分割する分割規則について一般的な表現で記載する。

【０１２１】

まず、元データ、乱数、分割データ、分割数および処理単位ビット長をそれぞれS,R,D,nおよびbで表すとともに、複数n個のうちの１つを表わす変数としてi(=1～n)およびj(=1～n 1)を用いて複数(n 1)個の元部分データ、複数(n 1)個の乱数部分データ、複数(n)個の分割データおよび各分割データの複数(n 1)個の分割部分データのそれぞれのうちの１つをそれぞれS(j),R(j),D(i)およびD(i,j)で表わす。

【０１２２】

それから、上記変数jを１からn 1まで変えて、各元部分データS(j)を元データSのb×(j 1)+1ビット目からbビット分のデータとして作成する。次に、U[n,n]をn×n行列である上三角行列とし、P[n,n]をn×n行列である回転行列としたとき、c(j,i,k)を(n 1)×(n 1)行列であるU[n 1,n 1]×P[n 1,n 1]^(j 1)のi行k列の値と定義する。そして、c(j,i,k)=1のとき、Q(j,i,k)=R(k)，c(j,i,k)=0のとき、Q(j,i,k)=0と定義したとき、変数iを１からnまで変えながら、各変数iにおいて変数jを１からn 1まで変えた場合において、
i＜nのとき、各分割部分データD(i,j)を

【数１４】

$$D(i,j) = S(j)*\left(\prod_{k=1}^{n-1} Q(j,i,k)\right)$$

【０１２３】

と設定し、またi=nのとき、各分割部分データD(i,j)を

D(i,j)=R(j)

と設定する。上記処理を元データSの先頭から末尾まで繰り返し行うことにより元データSから分割数nの分割データを生成することができる。

【０１２４】

次に、上述したように元データSを４分割して生成された分割データD(1),D(2),D(3),D(4)から元データSを復元する処理について図６を参照して説明する。なお、図６に示す４

分割の場合には、変数jを3×m+1(m≧0である任意の整数）として、同図に示す一般的な定義式から次に示すように元データSを生成することができる。

【０１２５】

まず、分割データD(1),D(2)から元データSを求める場合について説明する。

【０１２６】

D(1,j)*D(2,j)=(S(j)*R(j)*R(j+1)*R(j+2))*((S(j)*R(j)*R(j+1))
=(S(j)*S(j))*(R(j)*R(j))*(R(j+1)*R(j+1))*R(j+2)
=0*0*0*R(j+2)
=R(j+2)

従って、D(1,j)*D(2,j)を計算すれば、乱数R(j+2)が求まり、同様にD(1,j+1)*D(2,j+1)を計算すれば、乱数R(j)が求まり、同様にD(1,j+2)*D(2,j+2)を計算すれば、乱数R(j+1)が求まり、これらの得られた乱数R(j),R(j+1),R(j+2)を用いれば、

D(1,j)*R(j)*R(j+1)*R(j+2)
=(S(j)*R(j)*R(j+1)*R(j+2))*(R(j)*R(j+1)*R(j+2))
=S(j)*(R(j)*R(j))*(R(j+1)*R(j+1))*(R(j+2)*R(j+2))
=S(j)*0*0*0
=S(j)

であるから、D(1,j)*R(j)*R(j+1)*R(j+2)を計算して、S(j)を求めてもよいし、D(2,j)*R(j)*R(j+1)からS(j)を求めることもできる。

【０１２７】

同様に、D(1,j+1)*R(j)*R(j+1)*R(j+2)またはD(2,j+1)*R(j+1)*R(j+2)を計算してS(j+1)を求めることができ、また同様にD(1,j+2)*R(j)*R(j+1)*R(j+2)またはD(2,j+2)*R(j)*R(j+2)を計算してS(j+2)を求めることができる。

【０１２８】

更に、上述したと同様に、D(2)とD(3)からSを求めることができる。

【０１２９】

具体的には、まずR(j),R(j+1),R(j+2)を求めてから、D(2,j),D(2,j+1),D(2,j+2)またはD(3,j),D(3,j+1),D(3,j+2)とR(j),R(j+1),R(j+2)のXOR演算によりS(j),S(j+1),S(j+2)を求めることができる。

【０１３０】

また更に、D(1)とD(4)またはD(2)とD(4)またはD(3)とD(4)からSを求めることができる。

【０１３１】

D(4)はRをそのものから定義したものであるから、計算することなくD(4)からR(j),R(j+1),R(j+2)を取得することができ、例えば、D(1,j),D(1,j+1),D(1,j+2)とR(j),R(j+1),R(j+2)のXOR演算によりS(j),S(j+1),S(j+2)を求めることができる。

【０１３２】

上述したように、演算回数の差が１である任意の２つの分割データD(1)とD(2)、または、D(2)とD(3)、または、D(4)と任意の１つの分割データD(1)またはD(2)またはD(3)からSが復元可能である。すなわち、４つの分割データの中から任意に３つの分割データを取得すれば、その中には必ず上述したいずれかのケースが含まれるため、４つのうち任意の３つの分割データから元データを復元可能である。

【０１３３】

図７は、５分割の場合の分割データと定義式を示す表である。この５分割の場合は、jを4×m+1(mはm≧0である任意の整数）として、分割データの定義式から、上述した４分割の場合の復元処理と同様のことが言える。従って、演算回数の差が１である任意の２つの分割データD(1)とD(2)、または、D(2)とD(3)、または、D(3)とD(4)、または、D(5)と任意の１つの分割データD(1)またはD(2)またはD(3)またはD(4)から元データSが復元可能である。そして、５つの分割データの中から任意に３つの分割データを取得すれば、その中には必ずこのいずれかのケースが含まれるため、５つのうち任意の３つから復元可能である

といえる。

【０１３４】

また、分割数nを５より大きくとった場合も同様にして分割データを構成すれば、nが奇数である場合は(n+1)/2個、nが偶数である場合は(n/2)+1個の分割データから元データを復元することができる。この個数は、n個の分割データがあったときに、隣り合ったものを選択せず、かつ、n個目の分割データを選択しないような最大個数に１を加えたものである。つまり、前記最大個数に１を加えれば演算回数の差が１である２つの分割データまたはn個目の分割データとその他のデータを必ず含むこととなるため、復元に必要な個数が前記のとおりといえる。

【０１３５】

次に、図８に示すフローチャートを参照して、分割数がnで、処理単位ビット長がbである場合の一般的な分割処理について説明する。

【０１３６】

まず、利用者は端末５から分割装置１にアクセスして元データSを送信し、分割装置１ではデータ送受信手段１７が端末５からの元データSを受信し、分割装置１に供給する（ステップＳ４０１）。また、利用者は端末５から分割数n(n≧3である任意の整数）を分割装置１に指示する（ステップＳ４０３）。この分割数nは分割装置１において予め定められた値を用いてもよい。処理単位ビット長bを決定する（ステップＳ４０５）。なお、bは０より大きい任意の整数である。次に、元データSのビット長がb×(n 1)の整数倍であるか否かを判定し、整数倍でない場合には、元データSの末尾を０で埋める（ステップＳ４０７）。また、整数倍を意味する変数mを０に設定する（ステップＳ４０９）。

【０１３７】

次に、元データSのb×(n 1)×m+1ビット目からb×(n 1)ビット分のデータが存在するか否かが判定される（ステップＳ４１１）。この判定の結果、データが存在しない場合は、ステップＳ４２１に進むことになるが、今の場合は、ステップＳ４０９で変数mは０に設定された場合であるので、データが存在するため、ステップＳ４１３に進む。

【０１３８】

ステップＳ４１３では、変数jを１からn 1まで変えて、元データSのb×((n 1)×m+j 1)+1ビット目からbビット分のデータを元部分データS((n 1)×m+j)に設定する処理を繰り返し、これにより元データSを処理単位ビット長bで区分けした(n 1)個の元部分データS(1),S(2), S(n 1)が生成される。

【０１３９】

次に、変数jを１からn 1まで変えて、乱数部分データR((n 1)×m+j)に乱数発生手段１５から発生する処理単位ビット長bの乱数を設定し、これにより乱数Rを処理単位ビット長bで区分けしたn 1個の乱数部分データR(1),R(2), R(n 1)が生成される（ステップＳ４１５）。

【０１４０】

次に、ステップＳ４１７において、変数iを１からnまで変えるとともに、更に各変数iにおいて変数jを１からn 1まで変えながら、ステップＳ４１７に示す分割データを生成するための定義式により複数の分割データD(i)の各々を構成する各分割部分データD(i,(n 1)×m+j)を生成する。この結果、次に示すような分割データDが生成される。

【０１４１】

分割データD
=n個の分割データD(i)=D(1),D(2), D(n)
第１の分割データD(1)
=n 1個の分割部分データD(1,j)=D(1,1),D(1,2), D(1,n 1)
第２の分割データD(2)
=n 1個の分割部分データD(2,j)=D(2,1),D(2,2), D(2,n 1)

第nの分割データD(n)

=n 1個の分割部分データD(n,j)=D(n,1),D(n,2), D(n,n 1)

このように変数m=0の場合について分割データDを生成した後、次に変数mを１増やし（ステップＳ４１９）、ステップＳ４１１に戻り、変数m=1に該当する元データSのb×(n 1)ビット以降について同様の分割処理を行う。最後にステップＳ４１１の判定の結果、元データSにデータがなくなった場合、ステップＳ４１１からステップＳ４２１に進み、上述したように生成した分割データDを分割装置１のデータ送受信手段１７からネットワーク３を介して保管サーバ７にそれぞれ送信し、各保管サーバ７に保管し、分割処理を終了する。図１では保管サーバは３個であるが、分割数に応じて保管サーバを増やし、各分割データを異なる保管サーバに保管することが望ましい。

【０１４２】

次に、図９に示すフローチャートを参照して、分割数nが２の場合の分割処理について説明する。すなわち、上述した各実施形態は図８のフローチャートのステップＳ４０３に示したように分割数nが３以上(n≧3)の場合についてのものであるので、図９を用いて分割数nが２の場合について説明する。

【０１４３】

まず、利用者は端末５から分割装置１にアクセスして元データSを分割装置１に供給する（ステップＳ５０１）。また、利用者は端末５から分割数nとして２を分割装置１に指示する（ステップＳ５０３）。この分割数nは分割装置１において予め定められた値を用いてもよい。それから処理単位ビット長bとして８ビットを決定する（ステップＳ５０５）。次に、元データSのビット長が８の整数倍であるか否かを判定し、整数倍でない場合には、元データSの末尾を０で埋める（ステップＳ５０７）。また、整数倍を意味する変数mを０に設定する（ステップＳ５０９）。

【０１４４】

次に、元データSの8×m+1ビット目から８ビット分のデータが存在するか否かが判定される（ステップＳ５１１）。この判定の結果、データが存在しない場合は、ステップＳ５２１に進むことになるが、今の場合は、変数mは０に設定されているので、データが存在するため、ステップＳ５１３に進む。

【０１４５】

ステップＳ５１３では、元データSの8×m+1ビット目から８ビット分のデータを元部分データS(m+1)に設定し、これにより元部分データS(1)が生成される。

【０１４６】

次に、乱数部分データR(m+1)に乱数発生手段１５から発生する８ビットの乱数を設定し、これにより乱数部分データR(1)が生成される（ステップＳ５１５）。

【０１４７】

次に、ステップＳ５１７において、同ステップに示す定義式により分割データDの各々を構成する各分割データD(1,m+1),D(2,m+1)が生成される。

【０１４８】

このように変数m=0の場合について分割データDを生成した後、次に変数mを１増やし（ステップＳ５１９）、ステップＳ５１１に戻り、変数m=1に該当する元データSの８ビット以降について同様の分割処理を行う。最後にステップＳ５１１の判定の結果、元データSにデータがなくなった場合、ステップＳ５１１からステップＳ５２１に進み、上述したように生成した分割データD(1)からD(2)を分割装置１のデータ送受信手段１７からネットワーク３を介して保管サーバ７にそれぞれ送信し、各保管サーバ７に保管し、分割処理を終了する。図１では保管サーバは３個であるが、このうち２個の保管サーバに各分割データを保管すればよい。

【０１４９】

ここにおいて、上述した図９のフローチャートのステップＳ５１７における定義式による分割データの生成処理、具体的には分割数n=2の場合の分割データの生成処理について詳しく説明する。

【０１５０】
変数m=0の場合には、ステップＳ５１７に示す定義式から各分割データD(1,1),D(2,1)は、次のようになる。
【０１５１】
D(1,1)=S(1)*Q(1,1,1)
D(2,1)=R(1)
次に、Q(j,i,k)を具体的に求める。ここで、n=2を定義に当てはめると、j,i,kはいずれも１しか値をとらない。
【０１５２】
c(j,i,k)は1×1行列であるU[1,1]×(P[1,1])^(j 1)のi行k列の値としたとき下記のように定義される。
【０１５３】
c(j,i,k)=1 のとき Q(j,i,k)=R(k)
c(j,i,k)=0 のとき Q(j,i,k)=0
U[1,1]×(P[1,1])^(j 1)=U[1,1]×(P[1,1])^0
=(1)×E[1,1]
=(1)×(1)
=(1)
従って、c(1,1,1)は１であるから、Q(1,1,1)はR(1)と定義される。
【０１５４】
以上から定義式は
D(1,1)=S(1)*R(1)
D(2,1)=R(1)
となる。変数mを使用した形式では、
D(1,m+1)=S(m+1)*R(m+1)
D(2,m+1)=R(m+1)
となる。
【０１５５】
なお、分割数n=2の場合には、２個の分割データのうち、どちらか一方を取得しただけでは、元データSを復元することはできず、２個のすべての分割データを取得して元データSを復元することになる。
【０１５６】
さて、上述した実施形態においては、ここの分割データのみから、それを構成する部分データ間の演算を行うことによって乱数成分が失われる場合がある。即ち、例えば３分割の場合、各分割部分データは次のように定義される。
【０１５７】
D(1,1)=S(1)*R(1)*R(2), D(1,2)=S(2)*R(1)*R(2),
D(2,1)=S(1)*R(1), D(2,2)=S(2)*R(2),
D(3,1)=R(1), D(3,2)=R(2),
D(1)について見ると、例えば、D(1,1)、D(1,2)が取得できると、
D(1,1)*D(1,2)=(S(1)*R(1)*R(2))*(S(2)*R(1)*R(2))
=S(1)*S(2)*((R(1)*R(1))*((R(2)*R(2))
=S(1)*S(2)*0*0
=S(1)*S(2)
となる。一般にはD(1,j)*D(1,j+1)=S(j)*S(j+1)である。ここでｊはｊ＝２×ｍ＋１、ｍはｍ≧０の任意の整数である。
【０１５８】
D(1,1)、D(1,2)は、上記の定義より、、元データと乱数の演算により生成されたものであり、D(1,1)、D(1,2)それぞれを見ても元データの内容は分からないが、D(1,1)＊D(1,2)の演算を行うことによりS(1)*S(2)が算出される。これは元データそのものではないが、

乱数成分を含んでいない。

【０１５９】

このように乱数成分が失われると、個々の元部分データについて、例えばS(2)の一部が既知である場合にはS(1)の一部が復元可能となるので、安全ではないと考えられる。例えば、元データが標準化されたデータフォーマットに従ったデータであって、S(2)がそのデータフォーマット中のヘッダ情報やパディング（例えば、データ領域の一部を０で埋めたもの）などを含む部分であった場合には、これらのデータフォーマット固有のキーワードや固定文字列などを含むため、その内容は予測され得る。また、S(2)のうち既知の部分とS(1)*S(2)の値から、S(1)の一部が復元可能である。

【０１６０】

４分割の場合は、図６より、D(2,j)*D(2,j+1)*D(2,j+2)=S(j)*S(j+1)*S(j+2)である。ここでｊはｊ＝８×ｍ＋１、ｍはｍ≧０の任意の整数である。

【０１６１】

５分割の場合は、図７より、D(i,j)*D(i,j+1)*D(i,j+2)*D(i,j+3)=S(j)*S(j+1)*S(j+2)*S(j+3)である。ここでiは１または８，ｊはｊ＝４×ｍ＋１、ｍはｍ≧０の任意の整数である。

【０１６２】

分割数が５より大きい場合も同様に演算により、乱数成分が失われる。なお、分割数が２の場合にはこのような、問題は生じない。

【０１６３】

この問題を解決する一つの方法は以下の通りである。これは８分割の場合に適用可能な方法である。図１０におけるD(1,j+1)とD(2,j+1)は、図４におけるD(1,j+1)とD(2,j+1)を入れ替えたものである。ここでｊはｊ＝２×ｍ＋１、ｍはｍ≧０の任意の整数である。

【０１６４】

この場合、個々の分割データのみでは、それを構成する分割部分データ間で演算を行っても乱数成分が失われない。これは、図１０より

D(1,j)*D(1,j+1)=(S(j)*R(j)*R(j+1))*(S(j+1)*R(j+1))
=S(j)*S(j+1)*R(j)*((R(j+1)*R(j+1))
=S(j)*S(j+1)*R(j)*0
=S(j)*S(j+1)*R(j)

D(2,j)*D(2,j+1)=(S(j)*R(j))*(S(j+1)*R(j)*R(j+1))
=S(j)*S(j+1)*(R(j)*R(j))*R(j+1))
=S(j)*S(j+1)*0*R(j+1)
=S(j)*S(j+1)*R(j+1)

D(8,j)*D(8,j+1)=R(j)*R(j+1)

となるからである。

【０１６５】

また、この場合、８つの分割データのうち２つから、元データを復元することができるという特性は失われていない。これは、D(1)、D(2)を取得してＳを復元する場合には、図１０におけるD(1)、D(2)は、図４におけるD(1)、D(2)を構成する分割部分データを入れ替えたものにすぎないので、明らかにこれらから元データを復元することができ、また、D(1)とD(8)またはD(2)とD(8)を取得してＳを復元する場合には、D(8)は乱数のみからなる分割データであるので、D(1)またはD(2)の分割部分データ毎に必要な個数の乱数との排他的論理和演算を行うことにより、乱数部分を消去して元データを復元することができるからである。

【０１６６】

上記の問題を解決する別の方法は以下の通りである。これは分割数が８以上である場合に分割数には関係なく適用可能な方法である。図１１、図１２、図１３は、図４、図６、図７においてD(i,j)を生成する個々の定義式からＲ(j)を削除したものである。ここでiはn-1>i>0であり、ｊはｊ＝(n-1)×ｍ＋１、ｍはｍ≧０の任意の整数、nは分割数である。

これは分割数が５より大きい場合でも同様である。

【０１６７】

この場合も、個々の分割データのみでは、それを構成する分割部分データ間で演算を行っても乱数成分が失われない。これは、図４、図６、図７においては、個々の分割データの分割部分データ間で演算をすると乱数部分が消去されて、３分割の場合には D(1,j)*D(1,j+1)=S(j)*S(j+1)（ｊはｊ＝2×ｍ＋１、ｍはｍ≧０の任意の整数）となり、４分割の場合にはD(2,j)*D(2,j+1)*D(2,j+2)=S(j)*(S(j+1)*S(j+2)（ｊはｊ＝3×ｍ＋１、ｍはｍ≧０の任意の整数）となり、5分割の場合にはD(i,j)*D(i,j+1)*D(i,j+2)*D(i,j+3)=S(j)*S(j+1)*S(j+2)*S(j+3)（iは１または３，ｊはｊ＝４×ｍ＋１、ｍはｍ≧０の任意の整数）となっていたが、上記の通りD(i,j)を生成する個々の定義式からR(j)を削除した（iは n 1>i>0、ｊはｊ＝(n 1)×ｍ＋１、ｍはｍ≧０の任意の整数、nは分割数）ので、一つのR(j)が消去されずに残ることになるためである。

【０１６８】

また、この場合も、ｎ個の分割データのうちの所定の個数の分割データから、元データを復元することができるという特性は失われていない。

【０１６９】

まず３分割の場合には、D(1)、D(2)を取得してＳを復元する場合は、上述した通りR(j)（ｊはｊ＝2×ｍ＋１、ｍはｍ≧０の任意の整数）がD(1,j)*D(2,j+1)から求まり、S(j)が

D(2,j)*R(j)=S(j)*R(j)*R(j)
=S(j)*0
=S(j)

から求まり、R(j+1)が

D(1,j)*S(j)=S(j)*R(j+1)*S(j)
=S(j)*S(j)*R(j+1)
=0*R(j+1)
=R(j+1)

から求まり、上述した通りS(j+1)がD(2,j+1)*R(j+1)から求まる。また、D(1)とD(3)またはD(2)とD(3)を取得してＳを復元する場合には、D(3)は乱数のみからなる分割データであるので、D(1)またはD(2)の分割部分データ毎に必要な個数の乱数との排他的論理和演算を行うことにより、乱数部分を消去して元データを復元することができる。

【０１７０】

次に４分割の場合には、D(1)、D(2)を取得してＳを復元する場合は、R(j+2)が

D(1,j)*D(2,j)=(S(j)*R(j+1)*R(j+2))*(S(j)*R(j+1))
=(S(j)*S(j))*(R(j+1)*R(j+1))*R(j+2)
=0*0*R(j+2)
=R(j+2)

から求まり、上述した通りS(j)がD(1,j)*R(j+1)*R(j+2)またはD(2,j)*R(j+1)から求まる。

【０１７１】

また、D(2)とD(3)を取得してＳを復元する場合には、上述した通りR(j+2)がD(2,j+1)*D(3,j+1)から求まり、R(j)がD(2,j+2)*D(3,j+2)から求まり、S(j)が

D(3,j)*R(j)=(S(j)*R(j))*R(j)
=S(j)*(R(j)*R(j))
=S(j)*0
=S(j)

から求まり、R(j+1)が

D(2,j)*S(j)=(S(j)*R(j+1))*S(j)
=(S(j)*S(j))*R(j+1)
=0*R(j+1)
=R(j+1)

から求まり、上述した通りS(j)がD(1,j)*R(j+1)*R(j+2)または D(2,j)*R(j+1)から求まる。

【０１７２】

また、D(4)と任意の一つの分割データD(1)またはD(2)またはD(3)を取得してＳを復元する場合には、D(4)は乱数のみからなる分割データであるので、D(1)またはD(2)またはD(3)の分割部分データ毎に必要な個数の乱数との排他的論理和演算を行うことにより、乱数部分を消去して元データを復元することができる。

【０１７３】

従って、演算回数の差が１である任意の２つの分割データD(1)とD(2)、または、D(2)とD(3)、または、D(4)と任意の１つの分割データD(1)またはD(2)またはD(3)からSが復元可能である。すなわち、４つの分割データの中から任意に３つの分割データを取得すれば、その中には必ず上述したいずれかのケースが含まれるため、４つのうち任意の３つの分割データから元データを復元可能である。

【０１７４】

次に５分割の場合については、D(1)とD(2)またはD(2)とD(3)を取得してＳを復元する場合、D(3)とD(4)を取得してＳを復元する場合、D(5)と任意の一つの分割データD(1)またはD(2)またはD(3)またはD(4)を取得してＳを復元する場合のいづれも４分割の場合と同様である。

【０１７５】

従って、演算回数の差が１である任意の２つの分割データD(1)とD(2)、または、D(2)とD(3)、または、D(3)とD(4)、または、D(5)と任意の１つの分割データD(1)またはD(2)またはD(3)またはD(4)から元データSが復元可能である。そして、５つの分割データの中から任意に３つの分割データを取得すれば、その中には必ずこのいずれかのケースが含まれるため、５つのうち任意の３つから復元可能であるといえる。

【０１７６】

また、分割数nを５より大きくとった場合も同様にして分割データを構成すれば、nが奇数である場合は(n+1)/2個、nが偶数である場合は(n/2)+1個の分割データから元データを復元することができる。この個数は、n個の分割データがあったときに、隣り合ったものを選択せず、かつ、n個目の分割データを選択しないような最大個数に１を加えたものである。つまり、前記最大個数に１を加えれば演算回数の差が１である２つの分割データまたはn個目の分割データとその他のデータを必ず含むこととなるため、復元に必要な個数が前記のとおりといえる。

【０１７７】

なお、上記実施形態のデータ分割方法の処理手順をプログラムとして例えばＣＤやＦＤなどの記録媒体に記録して、この記録媒体をコンピュータシステムに組み込んだり、または記録媒体に記録されたプログラムを通信回線を介してコンピュータシステムにダウンロードしたり、または記録媒体からインストールし、該プログラムでコンピュータシステムを作動させることにより、データ分割方法を実施するデータ分割装置として機能させることができることは勿論であり、このような記録媒体を用いることにより、その流通性を高めることができるものである。

【０１７８】

上述してきたように、本実施例によれば、所定の定義式が元部分データと乱数部分データの排他的論理和からなるので、従来のように多項式や剰余演算を行う高速かつ高性能な演算処理能力を必要とせず、大容量のデータに対しても簡単な演算処理を繰り返して分割データを簡単かつ迅速に生成することができる。

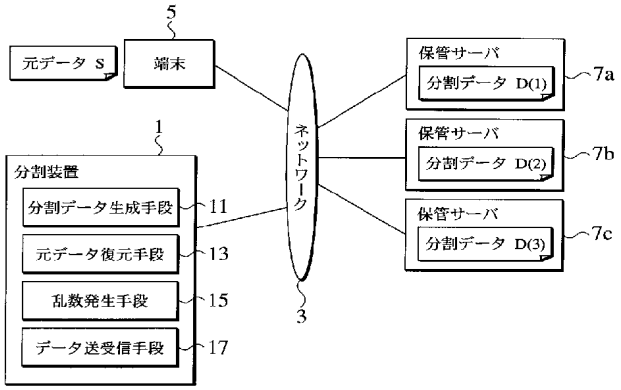
【０１７９】

また、生成した複数の分割データのうち分割数よりも少ない数の分割データに対して定義式を適用することにより元データを復元するので、分割数よりも少ない任意の数xの分割データで元データを復元でき、分割数からxを減算した数までの分割データを紛失したり破壊したとしても、元データを復元することができる。

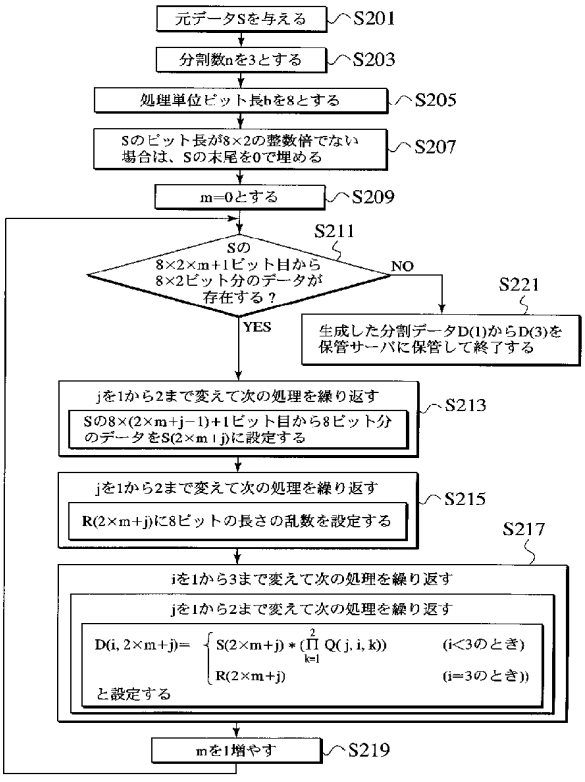
【０１８０】
さらに、元データをネットワークを介して端末から受信し、この元データに対して元部分データ、乱数部分データおよび分割部分データの生成処理を施して生成された複数の分割部分データをネットワークを介して保管サーバに送信して保管するので、多数のユーザが端末からネットワークを介してアクセスして分割処理を依頼することができ、共通化および経済化を図ることができる。
【０１８１】
なお、上述した実施形態は、分割データは、乱数のみからなる１つの分割データと、１つの元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる１つ以上の分割データを含む場合であるが、上述した実施形態を変形して分割データは、乱数のみからなる１つ以上の分割データと、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる１つ以上の分割データを含むものとしても良い。また、上述した実施形態を変形して、分割データは、１つ以上の元部分データと１つ以上の乱数部分データの排他的論理和演算によって生成された分割部分データからなる２つ以上の分割データを含むものとしても良い。
【図面の簡単な説明】
【０１８２】
【図１】本発明の一実施形態に係るデータ分割方法を実施するデータ分割装置を含むシステム構成図である。
【図２】図１に示す実施形態のデータ分割装置の分割数n=3の場合の分割処理を示すフローチャートである。
【図３】１６ビットの元データSを８ビットの処理単位ビット長に基づいて分割数n=3で3分割する場合の各データと定義式および各分割部分データから元データを復元する場合の計算式などを示す表である。
【図４】分割数n=3の場合の分割データ、分割部分データ、各分割部分データを生成する定義式を示す表である。
【図５】図１に示す実施形態のデータ分割装置の分割数n=4の場合の分割処理を示すフローチャートである。
【図６】分割数n=4の場合の分割データ、分割部分データ、各分割部分データを生成する定義式を示す表である。
【図７】分割数n=5の場合の分割データ、分割部分データ、各分割部分データを生成する定義式を示す表である。
【図８】図１に示す実施形態のデータ分割装置の分割数がnで処理単位ビット長がbである場合の一般的な分割処理を示すフローチャートである。
【図９】図１に示す実施形態のデータ分割装置の分割数が２である場合の分割処理を示すフローチャートである。
【図１０】分割数n=3の場合の分割データ、分割部分データ、各分割部分データを生成する定義式の別の例を示す表である。
【図１１】分割数n=3の場合の分割データ、分割部分データ、各分割部分データを生成する定義式の更に別の例を示す表である。
【図１２】分割数n=4の場合の分割データ、分割部分データ、各分割部分データを生成する定義式の別の例を示す表である。
【図１３】分割数n=5の場合の分割データ、分割部分データ、各分割部分データを生成する定義式の別の例を示す表である。
【符号の説明】
【０１８３】
１　分割装置
３　ネットワーク
５　端末

７ａ，７ｂ，７ｃ　保管サーバ
１１　分割データ生成手段
１３　元データ復元手段
１５　乱数発生手段
１７　データ送受信手段

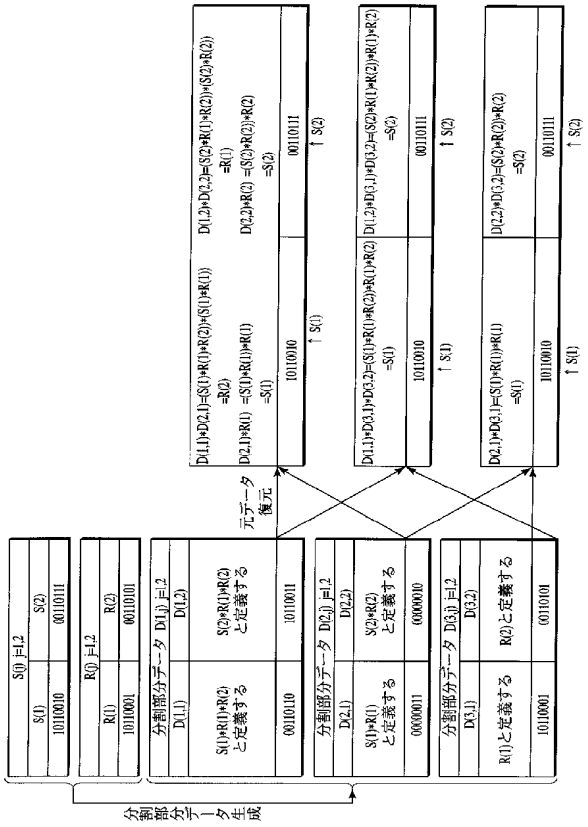
【図１】



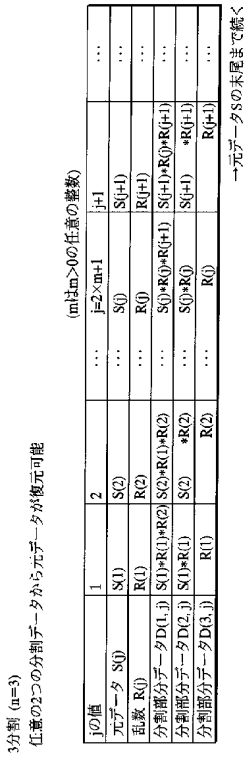
【図２】



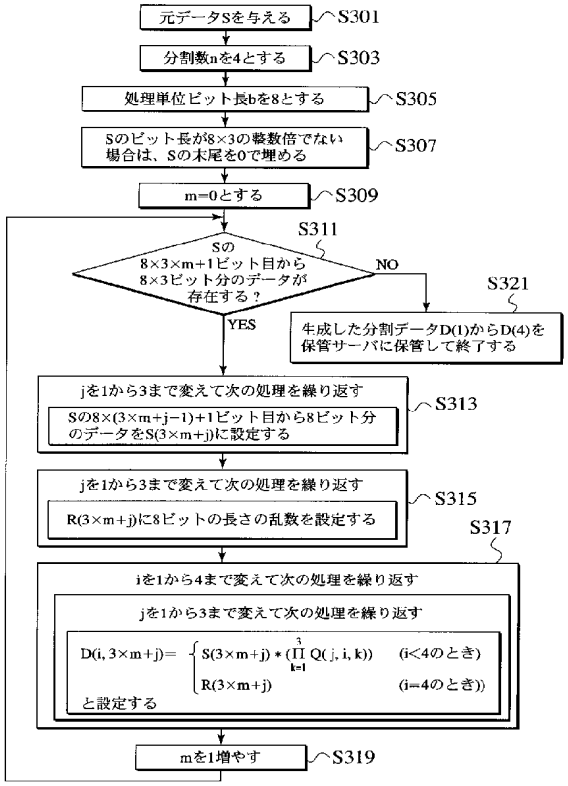
【図３】


S(0) j=1,2
S(1)
10110010
S(2)
00110111
R(0) j=1,2
R(1)
10110001
R(2)
00110101
分割部分データ D(1,j) j=1,2
D(1,1)
S(1)*R(1)+R(2)と定義する
D(1,2)
S(2)*R(1)*R(2)と定義する
00100110
10110011
分割部分データ D(2,j) j=1,2
D(2,1)
S(1)*R(1)と定義する
D(2,2)
S(2)*R(2)と定義する
00000011
00000000
分割部分データ D(3,j) j=1,2
D(3,1)
R(1)と定義する
D(3,2)
R(2)と定義する
10110001
00110101
D(1,1)*D(2,1)=(S(1)*R(1)+R(2))*(S(1)*R(1))=R(2)
D(1,2)*D(2,1)=(S(1)*R(1)+R(2))*(S(2)*R(2))=R(1)
D(2,2)*R(1)=(S(1)*R(1))*R(1)=S(1)
D(2,2)*R(2)=(S(2)*R(2))*R(2)=S(2)
10110010
00110111
↑S(1)
↑S(2)
元データ復元
D(1,1)*D(3,1)=D(3,2)=(S(1)*R(1)+R(2))*R(1)=R(2)
D(1,2)*D(3,1)=D(3,2)=(S(2)*R(1)*R(2))*R(1)=R(2)
=S(1)
=S(2)
10110010
00110111
↑S(1)
↑S(2)
D(2,1)*D(3,1)=(S(1)*R(1))*R(1)=S(1)
D(2,2)*D(3,2)=(S(2)*R(2))*R(2)=S(2)
10110010
00110111
↑S(1)
↑S(2)
分割結合データ生成

【図４】


3分割 (n=3)
任意の2つの分割データから元データが復元可能
(mはm>0の任意の整数)
jの値
元データ S(j)
乱数 R(j)
分割部分データD(1, j)
分割部分データD(2, j)
分割部分データD(3, j)
1
2
…
j=2×m+1
j+1
S(1)
S(2)
S(j)
S(j+1)
R(1)
R(2)
R(j)
R(j+1)
S(1)*R(1)+R(2)
S(2)*R(1)+R(2)
S(j)*R(j)+R(j+1)
S(j+1)*R(j)+R(j+1)
S(2)*R(1)
S(2) *R(2)
S(j)*R(j)
S(j+1) *R(j+1)
R(1)
R(2)
R(j)
R(j+1)
→元データSの末尾まで続く

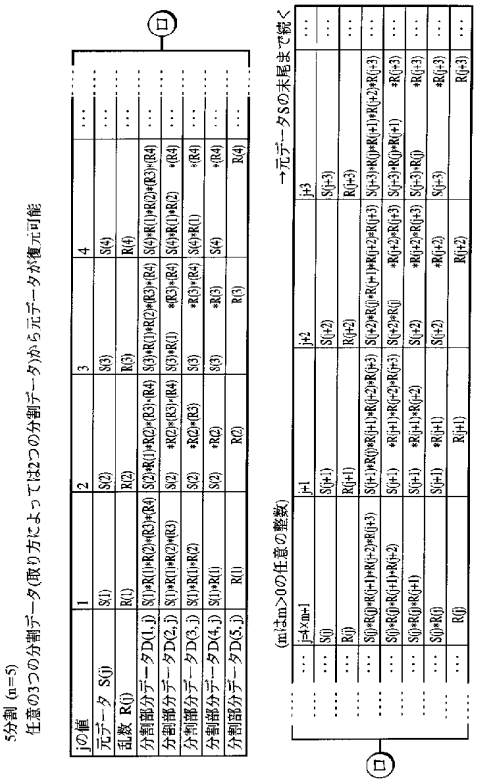
【図５】


元データSを与える
S301
分割数nを4とする
S303
処理単位ビット長bを8とする
S305
Sのビット長が8×3の整数倍でない場合は、Sの末尾を0で埋める
S307
m=0とする
S309
S311
Sの 8×3×m+1ビット目から 8×3ビット分のデータが 存在する？
NO
YES
S321
生成した分割データD(1)からD(4)を保管サーバに保管して終了する
jを1から3まで変えて次の処理を繰り返す
Sの8×(3×m+j−1)+1ビット目から8ビット分のデータをS(3×m+j)に設定する
S313
jを1から3まで変えて次の処理を繰り返す
R(3×m+j)に8ビットの長さの乱数を設定する
S315
S317
iを1から4まで変えて次の処理を繰り返す
jを1から3まで変えて次の処理を繰り返す
D(i, 3×m+j)= { S(3×m+j) * (∏ k=1 3 Q(j, i, k)) (i<4のとき)
R(3×m+j) (i=4のとき))
と設定する
mを1増やす
S319

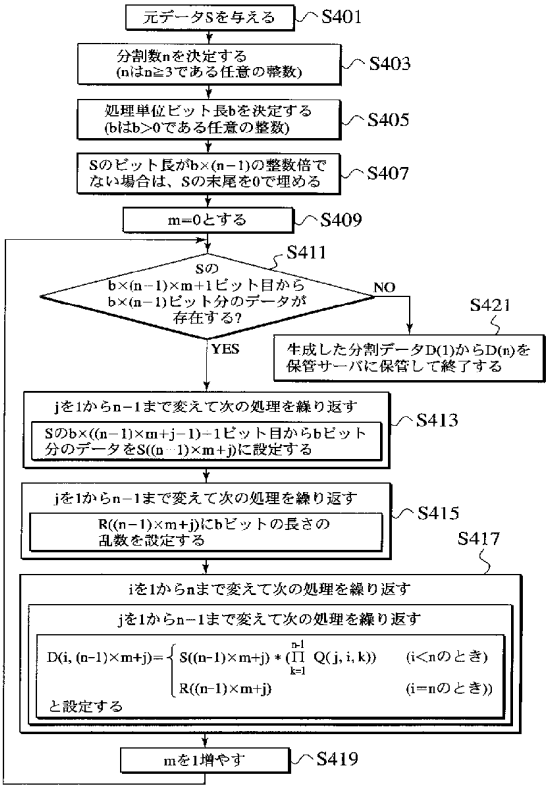
【図６】


4分割 (n=4)
任意の3つの分割データ取り方によっては2つの分割データから元データが復元可能
(mはm>0の任意の整数)
jの値
元データ S(j)
乱数 R(j)
分割部分データD(1, j)
分割部分データD(2, j)
分割部分データD(3, j)
分割部分データD(4, j)
1
2
3
…
j=3×m+1
j+1
j+2
S(1)
S(2)
S(3)
S(j)
S(j+1)
S(j+2)
R(1)
R(2)
R(3)
R(j)
R(j+1)
R(j+2)
S(1)*R(1)+R(2)*R(3)
S(2)*R(1)*R(2)*R(3)
S(3)*R(1)*R(2)*R(3)
S(j)*R(j)+R(j+1)*R(j+2)
S(j+1)*R(j)+R(j+1)*R(j+2)
S(j+2)*R(j)+R(j+1)*R(j+2)
S(1)*R(1)
S(2) *R(2)
S(3) *R(3)
S(j)*R(j)
S(j+1) *R(j+1)
S(j+2) *R(j+2)
R(1)
R(2)
R(3)
R(j)
R(j+1)
R(j+2)
→元データSの末尾まで続く

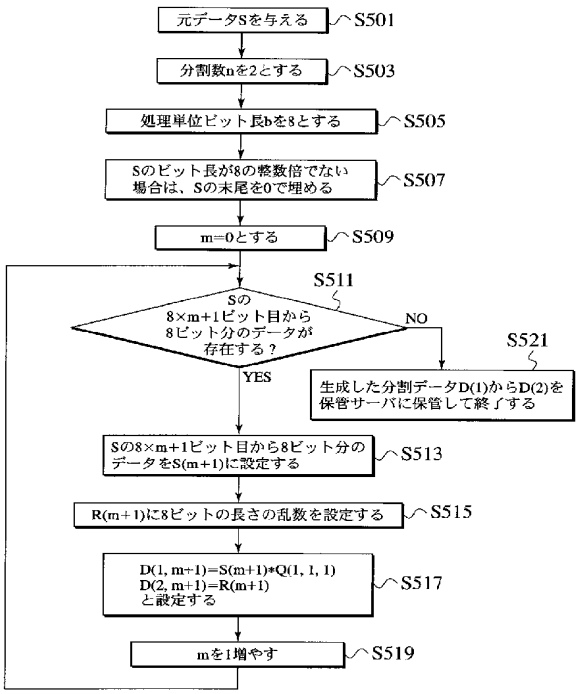
【図７】


5分割 (n=5)
任意の3つの分割データ(取り方によっては2つの分割データ)から元データが復元可能
jの値
元データ S(j)
乱数 R(j)
分割部分データD(1, j)
分割部分データD(2, j)
分割部分データD(3, j)
分割部分データD(4, j)
分割部分データD(5, j)
→元データSの末尾まで続く
(mは3m>0の任意の整数)

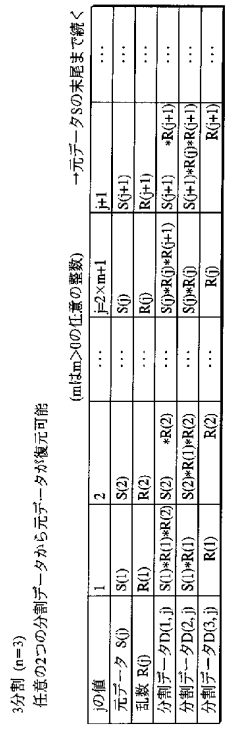
【図８】


元データSを与える
S401
分割数nを決定する
(nはn≧3である任意の整数)
S403
処理単位ビット長bを決定する
(bはb>0である任意の整数)
S405
Sのビット長がb×(n−1)の整数倍で
ない場合は、Sの末尾を0で埋める
S407
m=0とする
S409
S411
Sの
b×(n−1)×m+1ビット目から
b×(n−1)ビット分のデータが
存在する?
NO
YES
S421
生成した分割データD(1)からD(n)を
保管サーバに保管して終了する
jを1からn−1まで変えて次の処理を繰り返す
Sのb×((n−1)×m+j−1)−1ビット目からbビット
分のデータをS((n−1)×m+j)に設定する
S413
jを1からn−1まで変えて次の処理を繰り返す
R((n−1)×m+j)にbビットの長さの
乱数を設定する
S415
S417
iを1からnまで変えて次の処理を繰り返す
jを1からn−1まで変えて次の処理を繰り返す
D(i, (n−1)×m+j)= S((n−1)×m+j) * (∏ Q(j, i, k)) (i<nのとき)
R((n−1)×m+j) (i=nのとき))
と設定する
mを1増やす
S419

【図９】


元データSを与える
S501
分割数nを2とする
S503
処理単位ビット長bを8とする
S505
Sのビット長が8の整数倍でない
場合は、Sの末尾を0で埋める
S507
m=0とする
S509
S511
Sの
8×m+1ビット目から
8ビット分のデータが
存在する?
NO
YES
S521
生成した分割データD(1)からD(2)を
保管サーバに保管して終了する
Sの8×m+1ビット目から8ビット分の
データをS(m+1)に設定する
S513
R(m+1)に8ビットの長さの乱数を設定する
S515
D(1, m+1)=S(m+1)*Q(1, 1, 1)
D(2, m+1)=R(m+1)
と設定する
S517
mを1増やす
S519

【図１０】


3分割 (n=3)
任意の2つの分割データから元データが復元可能
jの値
元データ S(j)
乱数 R(j)
分割データD(1, j)
分割データD(2, j)
分割データD(3, j)
→元データSの末尾まで続く
(mは3m>0の任意の整数)

【図１１】

3分割 (n=3)
任意の2つの分割データから元データが復元可能

(mはm>0の任意の整数)

jの値	1	2	…	j=2×m+1	j+1	…
元データ S(j)	S(1)	S(2)	…	S(j)	S(j+1)	…
乱数 R(j)	R(1)	R(2)	…	R(j)	R(j+1)	…
分割データD(1, j)	S(1) *R(2)	S(2)×R(1)*R(2)	…	S(j) *R(j+1)	S(j+1)*R(j)*R(j+1)	…
分割データD(2, j)	S(1)*R(1)	S(2) *R(2)	…	S(j)*R(j)	S(j+1) *R(j+1)	…
分割データD(3, j)	R(1)	R(2)	…	R(j)	R(j+1)	…

→元データSの末尾まで続く

【図１２】

4分割 (n=4)
任意の3つの分割データ(取り方によっては2つの分割データ)から元データが復元可能

jの値	1	2	3	…
元データ S(j)	S(1)	S(2)	S(3)	…
乱数 R(j)	R(1)	R(2)	R(3)	…
分割データD(1, j)	S(1) *R(2)*(R3)	S(2)*R(1)*R(2)*(R3)	S(3)*R(1)*R(2)*(R3)	…
分割データD(2, j)	S(1) *R(2)	S(2) *R(2)*(R3)	S(3)*R(1) *(R3)	…
分割データD(3, j)	S(1)*R(1)	S(2) *R(2)	S(3) *R(3)	…
分割データD(4, j)	R(1)	R(2)	R(3)	…

(八)

(mはm>0の任意の整数)

→元データSの末尾まで続く

…	j=3×m+1	j+1	j+2	…
…	S(j)	S(j+1)	S(j+2)	…
…	R(j)	R(j+1)	R(j+2)	…
…	S(j) *R(j+1)*R(j+2)	S(j+1)*R(j)*R(j+1)*R(j+2)	S(j+2)*R(j)*R(j+1)*R(j+2)	…
…	S(j) *R(j+1)	S(j+1) *R(j+1)*R(j+2)	S(j+2)*R(j) *R(j+2)	…
…	S(j)*R(j)	S(j+1) *R(j+1)	S(j+2) *R(j+2)	…
…	R(j)	R(j+1)	R(j+2)	…

(八)

【図１３】

5分割 (n=5)
任意の3つの分割データ(取り方によっては2つの分割データ)から元データが復元可能

jの値	1	2	3	4	…
元データ S(j)	S(1)	S(2)	S(3)	S(4)	…
乱数 R (j)	R(1)	R(2)	R(3)	R(4)	…
分割データD(1, j)	S(1) *R(2)*(R3)*(R4)	S(2)*R(1)*R(2)*(R3)*(R4)	S(3)*R(1)*R(2)*R(3)*(R4)	S(4)*R(1)*R(2)*(R3)*(R4)	…
分割データD(2, j)	S(1) *R(2)*(R3)	S(2) *R(2)*(R3)*(R4)	S(3)*R(1) *(R3)*(R4)	S(4)*R(1)*R(2) *(R4)	…
分割データD(3, j)	S(1) *R(2)	S(2) *R(2)*(R3)	S(3) *R(3)*(R4)	S(4)*R(1) *(R4)	…
分割データD(4, j)	S(1)*R(1)	S(2) *R(2)	S(3) *R(3)	S(4) *(R4)	…
分割データD(5, j)	R(1)	R(2)	R(3)	R(4)	…

(二)

(mはm>0の任意の整数)

→元データSの末尾まで続く

…	j=4×m+1	j+1	j+2	j+3	…
…	S(j)	S(j+1)	S(j+2)	S(j+3)	…
…	R(j)	R(j+1)	R(j+2)	R(j+3)	…
…	S(j) *R(j+1)*R(j+2)*R(j+3)	S(j+1)*R(j)*R(j+1)*R(j+2)*R(j+3)	S(j+2)*R(j)*R(j+1)*R(j+2)*R(j+3)	S(j+3)*R(j)*R(j+1)*R(j+2)*R(j+3)	…
…	S(j) *R(j+1)*R(j+2)	S(j+1) *R(j+1)*R(j+2)*R(j+3)	S(j+2)*R(j) *R(j+2)*R(j+3)	S(j+3)*R(j)*R(j+1) *R(j+3)	…
…	S(j) *R(j+1)	S(j+1) *R(j+1)*R(j+2)	S(j+2) *R(j+2)*R(j+3)	S(j+3)*R(j) *R(j+3)	…
…	S(j)*R(j)	S(j+1) *R(j+1)	S(j+2) *R(j+2)	S(j+3) *R(j+3)	…
…	R(j)	R(j+1)	R(j+2)	R(j+3)	…

(二)

フロントページの続き

(72)発明者　荻原　利彦
東京都千代田区内幸町一丁目１番６号　エヌ・ティ・ティ・コミュニケーションズ株式会社内
(72)発明者　野村　進
東京都千代田区内幸町一丁目１番６号　エヌ・ティ・ティ・コミュニケーションズ株式会社内

Ｆターム(参考) 5B017 AA03　BA10　CA15　CA16